Personal Computer

DOS User's Manual

Personal Computer

DOS User's Manual

Notice

Information in this document is subject to change without notice and does not represent a commitment on the part of the Microsoft Corporation. The software described in this document is furnished under a license agreement or non-disclosure agreement. The software may be used or copied only in accordance with the terms of that agreement. It is against the law to copy MS-DOS or GW-BASIC on magnetic tape, disk, or any other medium for any purpose other than the purchaser's personal use.

Copyright (C) Microsoft Corporation, 1983,1984

Microsoft is a registered trademark, and MS and GW are trademarks of Microsoft Corporation.

IBM is a registered trademark of IBM Corporation.

CONTENTS

Chapter 1 INTRODUCTION

1.1 What is MS-DOS? ..... 1 - 1
1.2 What is an Operating System? ..... 1 - 1
1.3 Why is MS-DOS So Important? ..... 1 - 2
1.4 Features of DOS Version 3.00 ..... 1 - 2
1.5 About This Manual ..... 1 - 4
1.6 Syntax Notation ..... 1 - 5
1.7 MS-DOS Files ..... 1 - 6

Chapter 2 GETTING STARTED

2.1 What Happens When You First Load MS-DOS? ..... 2 - 1
2.2 How to Enter the Data and Time ..... 2 - 1
2.3 How to Change the Default Drive ..... 2 - 3
2.4 How to Format Your Disks ..... 2 - 4
2.5 How to Back Up Your Disks ..... 2 - 6
2.6 Automatic Program Execution ..... 2 - 6
2.7 Files ..... 2 - 8
2.8 How to Turn the System Off ..... 2 - 12

Chapter 3 MORE ABOUT FILES

3.1 How to Name Your Files ..... 3 - 1
3.2 Wild Cards ..... 3 - 2
3.3 Illegal Filenames ..... 3 - 4
3.4 How to Copy Your Files ..... 3 - 5
3.5 How to Protect Your Files ..... 3 - 6
3.6 Directories ..... 3 - 7
3.7 Filenames and Paths ..... 3 - 10

Chapter 4 LEARNING ABOUT COMMANDS

4.1 Introduction ..... 4 - 1
4.2 Types of MS-DOS Commands ..... 4 - 1
4.3 Command Options ..... 4 - 3
4.4 Information Common to All MS-DOS Commands ..... 4 - 4
4.5 Batch Processing ..... 4 - 5
4.6 The AUTOEXEC.BAT File ..... 4 - 7
4.7 Creating a.BAT File With Replaceable Parameters ..... 4 - 10
4.8 Input and Output ..... 4 - 12

Chapter 5 MS-DOS COMMANDS

5.1 Command Formats ..... 5 - 1
5.2 Summary of MS-DOS Commands ..... 5 - 1
5.3 Description of Commands ..... 5 - 4
ASSIGN ..... 5 - 4
ATTRIB ..... 5 - 5
BACKUP ..... 5 - 6
BREAK ..... 5 - 8
CHDIR ..... 5 - 9
CHKDSK ..... 5 - 10
CLS ..... 5 - 13
COMP ..... 5 - 14
COPY ..... 5 - 16
CTTY ..... 5 - 21
DATE ..... 5 - 22
DEL ..... 5 - 24
DIR ..... 5 - 25
DISKCOMP ..... 5 - 27
DISKCOPY ..... 5 - 29
EXE2BIN ..... 5 - 32
EXIT ..... 5 - 35
FIND ..... 5 - 36
FORMAT ..... 5 - 38
GRAPHICS ..... 5 - 46
JOIN ..... 5 - 47
LABEL ..... 5 - 49
MKDIR ..... 5 - 51
MODE ..... 5 - 52
MORE ..... 5 - 55
PARK ..... 5 - 56
PATH ..... 5 - 58
PRINT ..... 5 - 59
PROMPT ..... 5 - 63
RECOVER ..... 5 - 65
REM ..... 5 - 66
REN ..... 5 - 67
RESTORE ..... 5 - 68
RMDIR ..... 5 - 70
SET ..... 5 - 71
SHARE ..... 5 - 72
SORT ..... 5 - 74
SUBST ..... 5 - 76
SYS ..... 5 - 78
TIME ..... 5 - 80
TREE ..... 5 - 82

TYPE ..... 5 - 83
VER ..... 5 - 84
VERIFY ..... 5 - 85
VOL ..... 5 - 86
5.4 Batch Processing Commands ..... 5 - 87
ECHO ..... 5 - 88
FOR ..... 5 - 89
GOTO ..... 5 - 90
IF ..... 5 - 91
PAUSE ..... 5 - 92
SHIFT ..... 5 - 93

Chapter 6 MS-DOS EDITING AND FUNCTION KEYS
6.1 Special MS-DOS Editing Keys ..... 6 - 1
6.2 Control Character Functions ..... 6 - 5

Chapter 7 THE LINE EDITOR (EDLIN)
7.1 Introcution ..... 7 - 1
7.2 How to Start EDLIN ..... 7 - 1
7.3 Special Editing Keys ..... 7 - 3
7.4 Command Information ..... 7 - 15
7.5 EDLIN Commands ..... 7 - 18
7.6 Error Messages ..... 7 - 45

Chapter 8 THE LINKER PROGRAM (MS-LINK)
8.1 Introduction ..... 8 - 1
8.2 Overview of MS-LINK ..... 8 - 1
8.3 Definitions You'll Need to Know ..... 8 - 3
8.4 Files That MS-LINK Uses ..... 8 - 5
8.5 How to Start MS-LINK ..... 8 - 7
8.6 Command Characters ..... 8 - 11
8.7 Command Prompts ..... 8 - 12
8.8 MS-LINK Switches ..... 8 - 14
8.9 Sample MS-LiNK Session ..... 8 - 18
8.10 Error Messages ..... 8 - 20

Chapter 9 THE DEBUGGER PROGRAM (DEBUG)
9.1 Introduction ..... 9 - 1
9.2 How to Start DEBUG ..... 9 - 1
9.3 Command Information ..... 9 - 3
9.4 Parameters ..... 9 - 5
9.5 Error Messages ..... 9 - 39

Chapter 10 USING THE HARD DISK

10.1 Introduction ..... 10 - 1
10.2 Preparing Your Hard Disk ..... 10 - 1
10.3 Creating the MS-DOS Partition (Option 1) ..... 10 - 3
10.4 Activating the Startup Partition (Option 2) ..... 10 - 6
10.5 Deleting the MS-DOS Partition (Option 3) ..... 10 - 7
10.6 Displaying Partition Information (Option 4) ..... 10 - 8
10.7 Systems with Multiple Hard Disk Drivers (Option 5). 10 - 9

Appendix A Instructions for Users With Single-Drive Systems ..... A - 1
Appendix B Disk Errors ..... B - 1
Appendix C ANSI Escape Sequences ..... C - 1
Appendix D How to Configure Your System ..... D - 1

CHAPTER 1
INTRODUCTION

What Is MS-DOS?
What Is An Operating System?
Why Is MS-DOS So Important?
Features of DOS Version 3.00
About This Manual
Syntax Notation
MS-DOS Files

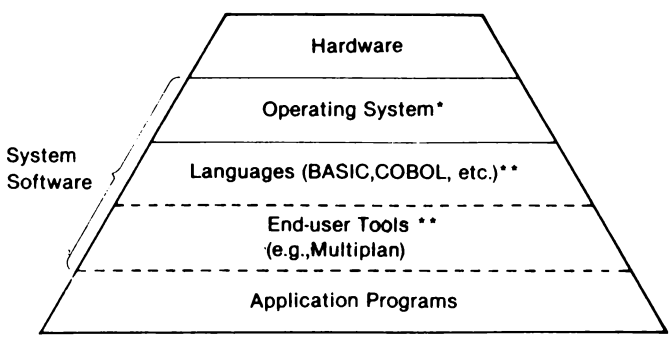
1.1 WHAT IS MS-DOS?

Microsoft® MS™-DOS is a disk operating system for 8086/8088-based computers. Through MS-DOS, you communicate with the computer, disk drives, and printer, managing these resources to your advantage.

1.2 WHAT IS AN OPERATING SYSTEM?

An operating system is your "silent partner" when you are using the computer. It provides the interface between the hardware and both you (the user) and the other system software. An operating system can be compared to the electricity in a house—you need it for the toaster and the blender to work, but you are not always aware that it's there.

An operating system (OS) is the piece of system software most closely associated with the hardware. The OS is unique to the microprocessor (computer). For example, MS-DOS runs on the 8086/8088 microprocessor family and will not run on another microprocessor (like the Z8000) unless major parts of the OS are rewritten. Figure 1 illustrates how the hardware, the system software and the application software are related.



* Must adapt to new hardware

** If adapted to operating system, these don't change

Figure 1. Hardware/Software Relationships

MS-DOS is a disk operating system that enables you to create and keep track of files, run and link programs, and access peripheral devices (for example, printers and disk drives) that are attached to your computer. MS-DOS is an important advance in microprocessor operating systems.

1.3 WHY IS MS-DOS SO IMPORTANT?

All Microsoft Languages (BASIC Interpreter, BASIC Compiler, FORTRAN, COBOL, Pascal) are available under MS-DOS. Users of MS-DOS are assured that their operating system will be the first that Microsoft will support when any new products or major releases are announced.

1.4 FEATURES OF DOS VERSION 3.00

This section will present the new features and changes of DOS 3.00 since the previous version of 2.10.

1.4.1 New Commands

The commands that follow have been added to the new version of DOS 3.00.

ATTRIB
The ATTRIB command is used for two functions. One is to mark a file as read-only. Meaning that you can read a file, but you cannot write to a file. The second is to display the attributes of a file(s).

LABEL
The LABEL command is used to give a disk a name which is referred to as a 'volume label'. When you list the directory of your disk the volume label is displayed and is also used to identify a disk. With the LABEL command you can change, add, or delete a volume label.

SHARE
The SHARE command is used to install file sharing support. With the installation of SHARE you can validate records in a file against file sharing rules.

DEVICE
By using the DEVICE configuration command a virtual disk can be installed. Virtual disks use a portion of your computer's memory to simulate a disk drive. They are very fast and can save time when application programs require access to a disk frequently.

FCBS
The number of file control blocks (FCBS) can be specified in your CONFIG.SYS file. These file control blocks will be concurrently open.

LASTDRIVE
The maximum number of disk drives that you can access can be specified in your CONFIG.SYS file.

1.4.2 Enhanced Commands

In DOS 3.00, the following commands have been enhanced. For a further description of their use and function please refer to Chapter 5 of this book.

External Commands
All external commands can be preceded by a drive and path. This will allow you to access external commands on directories other than the current directory.

FORMAT
The addition of the /4 switch gives FORMAT the ability to format single or double sided floppy disks in a high capacity drive.

BACKUP and RESTORE
The external commands of BACKUP and RESTORE can be used in the following media combinations.

Hard Disk ⟶ Floppy Disk
Hard Disk ⟶ Hard Disk
Floppy Disk ⟶ Hard Disk
Floppy Disk ⟶ Floppy Disk

DISKCOMP and DISKCOPY
High capacity floppy disks are now supported.

DATE
DATE can now be configured to the following formats:
yy-mm-dd, dd-mm-yy or mm-dd-yy.

1.5 ABOUT THIS MANUAL

This manual describes MS-DOS and how to use it. This chapter introduces some basic MS-DOS concepts; Chapter 2 discusses how to start using MS-DOS and how to format and back up your disks.

Chapter 3 tells you about files—what they are and how to use them. Chapters 4 through 6 introduce MS-DOS commands and Chapter 7 describes the line editor, EDLIN. Read these chapters carefully—they contain information on protecting your data, system commands, and the MS-DOS editing commands.

If you are writing programs and want to link separately-produced object modules and create relocatable modules, Chapter 8 describes a useful MS-DOS utility, MS-LINK.

Chapter 9 details about DEBUG utility. DEBUG eliminates the need to reassemble a program to see if a problem has been fixed by a minor change.

Chapter 10 guides you on the usage of FDISK utility. FDISK prepares and activates a hard disk.

Appendices to this manual include instruction for users with one-disk-drive systems and disk error messages.

1.6 SYNTAX NOTATION

The following syntax notation is used throughout this manual in descriptions of command and statement syntax:

[ ] Square brackets indicate that the enclosed entry is optional.

< > Angle brackets indicate data you must enter. When the angle brackets enclose lower case text, you must type in an entry defined by the text; for example, <*filename*>. When the angle brackets enclose upper case text, you must press the key named by the text; for example, <ENTER>

{ } Braces indicate that you have a choice between two or more entries. At least one of the entries enclosed in braces must be chosen unless the entries are also enclosed in square brackets.

Ellipses indicate that an entry may be repeated as many time as needed or desired.

| A bar indicates an OR statement in a command. When used with an MS-DOS filter, the bar indicates a pipe.

CAPS Capital letters indicate portions of statements or commands that must be entered, exactly as shown.

All other punctuation, such as commas, colons, slash marks, and equal signs, must be entered exactly as shown.

1.7 MS-DOS FILES

The MS-DOS disk contains the following files:

File Name	Function of File
ANSI.SYS	Supports ANSI escape sequence
ASSIGN.COM	Assigns physical disk drives
ATTRIB.EXE	Sets or displays attributes of a file
BACKUP.EXE	Backs up files
BASIC.COM	GW-BASIC system loader
BASICA.COM	GW-BASIC system loader
BASICA.EXE	GW-BASIC interpreter
CHKDSK.COM	Checks disks
COMMAND.COM	MS-DOS command processor
COMP.COM	Compares files
DEBUG.COM	Debugger
DISKCOMP.COM	Compares disks
DISKCOPY.COM	Back-up utility
EDLIN.COM	Line editor
EXE2BIN.EXE	Converts. EXE file
FDISK.COM	Prepares and activates hard disk
FIND.EXE	Finds a string in a list of files or standard input
FORMAT.COM	Formats disks
GRAPHICS.COM	Copies screen to printer
JOIN.EXE	Joins a disk drive to a specific pathname
LABEL.EXE	Gives a volume label to a disk
LINK.EXE	Linker
MODE.COM	Sets operation mode of peripherals
MORE.COM	Reviews text
PARK.COM	Repositions read/write head on the hard disk
PRINT.COM	Print spooler
RECOVER.COM	Recovers disks
RESTORE.EXE	Restores files
SHARE.EXE	Shares files
SORT.EXE	Sorts text
SUBST.EXE	Creates a string alias
SYS.COM	Transfers system
TREE.COM	Displays all directories
VDISK.SYS	Virtual disk driver
*MSDOS.SYS	MS-DOS operating system
*IO.SYS	Hardware-operating system interface

You will recognize this list of files when you have learned the DIR (Show Directory) command described in the next chapter. The two files preceded by an asterisk (*) are "hidden" files and will not appear when you enter a DIR command.

In the next chapter, you will learn how to start your MS-DOS system and how to format and back up your disks.

CHAPTER 2

GETTING STARTED

What Happens When You First Load MS-DOS ?
How To Enter The Date And Time
How To Change The Default Drive
How To Format Your Disks
  The FORMAT Command
How To Back Up Your Disks
  The DISKCOPY Command
Automatic Program Execution
Files
  What Is A File?
  How MS-DOS Keeps Track Of Your Files
  The DIR Command
  The CHKDSK Command
How To Turn The System Off

2.1 WHAT HAPPENS WHEN YOU FIRST LOAD MS-DOS?

Follow the separate Owner's Manual to insert and load the MS-DOS disk into your system. Loading MS-DOS takes from 10 to 30 seconds, depending on the size of memory in your computer.

Once MS-DOS has been loaded, the system searches the MS-DOS disk for the COMMAND.COM file and loads it into memory. The COMMAND.COM file is a program that processes the commands you enter and then runs the appropriate programs. It is also called the *command processor.*

When the command processor is loaded, you will see the following display on your screen (the underscore represents the cursor):

```
Current date is Tue 1-01-1985
Enter new date:_
```

Enter current date at your computer. How to enter the date will be mentioned in the next paragraph. When the date is entered, the next message will appear:

```
Current time is 0:00:11.64
Enter new time:_
```

You must now enter the current time at your computer.

2.2 HOW TO ENTER THE DATE AND TIME

Type today's date in an mm-dd-yy format, where:

mm is a one- or two-digit number from 1-12 (representing month)

dd is a one- or two-digit number from 1-31 (representing day of month)

yy is a two-digit number from 80-99 (the 19 is assumed), or a four-digit number from 1980-2099 (representing year)

Any date is acceptable in answer to the new date prompt as long as it follows the above format. Separators between the numbers can be hyphens (-) or slashes (/). For example:

6-1-82 or 06/01/82

are both acceptable answers to the Enter new date: prompt.

If you enter an invalid date or form of date, the system will prompt you again with Enter new date:.

After you respond to the new date prompt and enter your answer by pressing the <**ENTER**> key, you will see a prompt similar to this:

Current time is 0:00:33.88

Enter new time:_

Enter the current time in the hh:mm format, where:

hh is a one- or two-digit number from 0-23 (representing hours)

mm is a one- or two-digit number from 0-59 (representing minutes)

MS-DOS uses this time value to keep track of when you last updated and/or created files on the system. Notice that MS-DOS uses military time; for instance, 1:30 p.m is written 13:30

Example:

Current time is 0:00:33.88

Enter new time: 13:30

You should only use the colon (:) to separete hours and minutes. If you enter an invalid number separator, MS-DOS will repeat the prompt.

You have now completed the steps for starting MS-DOS.

2.3 HOW TO CHANGE THE DEFAULT DRIVE

After you have answered the new time prompt, a message is displayed that looks like this:

```
Microsoft MS-DOS
Version 3.10 (C) copyright Microsoft Corp 1981/1985
A>
```

The A> is the MS-DOS prompt from the command processor. It tells you that MS-DOS is ready to accept commands. If you have inserted the MS-DOS disk into a drive other than A:, the command processor prompt will reflect that drive (for example, B>). However,usually you will load MS-DOS in drive A:.

The A in the previous prompt represents the default disk drive. This means that MS-DOS will search only the disk in drive A: for any filenames you may enter and will write files to that disk unless you specify a different drive. You can ask MS-DOS to search the disk in drive B: by changing the drive designation or by specifying B: in a command. To change the disk drive designation, enter the new drive letter followed by a colon. For example:

```
A>      (MS-DOS prompt)
A>B:    (you have typed B: in response to the prompt)
B>      (system responds with B: and drive B: is now the
         default drive)
```

The system prompt B> will appear and MS-DOS will search only the disk in drive B: until you specify a different default drive.

If you have only one disk drive attached to your computer, turn to Appendix A, Instructions for Users with Single-Drive Systems, for important information. Your computer may have more than 2 disk drives; refer to the Owner's Manual on the use of these drives.

2.4 HOW TO FORMAT YOUR DISKS

You must "format" all new disks before they can be used by MS-DOS.

A blank disk must be formatted with the MS-DOS FORMAT command. The FORMAT command changes the disk to a format that MS-DOS can use; it also analyzes the disk for defective tracks. If the disk is not already blank, formatting it will destroy any data that exists on the disk. Although this can be a convenient way to make a blank disk, it is recommended that the MS-DOS DEL command be used for this purpose. Refer to Chapter 5,"MS-DOS Commands", for more information on the DELETE command.

2.4.1 The FORMAT Command

The syntax of the FORMAT command is :

FORMAT [*d:*]

where:

d: is the drive designation (the drive that contains the disk to be formatted)

Note that the brackets identify optional information. If you do not specify a disk drive (for example, A: or B:), MS-DOS will format the disk in the default drive.

With the MS-DOS disk already in drive A:, you are ready to format your new blank disk. The following command will format the new disk in drive B:.

FORMAT B:

MS-DOS issues the following message:

Insert new diskette for drive B:
and strike ENTER when ready

After you insert the new disk in drive B: and press keyboard, the system responds:

```
Formatting...
```

while MS-DOS is formatting your disk. If you add the /V switch to the FORMAT command, MS-DOS will ask for the volume label of the disk. (See below.) The /S switch tells MS-DOS to copy its system files onto the new disk.

When the formatting is finished, MS-DOS will issue a message similar to this:

```
Formatting...Format complete
System transferred

Volume label (11 characters. ENTER for none) ?
```

Volume labels are useful to identify disks—they are like a name tag for each disk. When you assign a unique volume label to a disk, you can always be sure that you know which disk you are using. The volume label you assign to a disk is displayed by issuing the MS-DOS VOL command (refer to Chapter 5,"MS-DOS Commands", for more information on the VOL command). Type a volume label in response to the above prompt if you want to identify this disk, and press <ENTER>. An example of a volume label is PROGRAMS. If you do not want to attach a label to this disk, simply press the <ENTER> key. You will see on your screen a message similar to this:

```
1213952 bytes total disk space
1213952 bytes available on disk

Format another (Y/N)?_
```

Type Y to format another disk. Type N to end the FORMAT program.

2.5 HOW TO BACK UP YOUR DISKS

It is strongly recommended that you make backup copies of all your disks. If a disk becomes damaged or if files are accidentally erased, you will still have all of the information on your backup disk. **You should make a backup copy of your MS-DOS disk also.** You can back up disks by using the MS-DOS DISKCOPY command. This command is described below.

2.5.1 The DISKCOPY Command

The DISKCOPY command copies the contents of a disk onto another disk. You can use this command to duplicate both the MS-DOS disk and a disk that contains your own files. DISKCOPY is the fastest way of copying a disk because it copies the entire disk in one operation, including MS-DOS system files if they exist.

The format of the DISKCOPY command is:

DISKCOPY [*drive1:*] [*drive2:*]

Drive1 is the disk drive that contains the disk that you want to copy; drive2 is the disk drive that contains the blank or "destination" disk. The blank disk must be formatted prior to running DISKCOPY.

For example, if you want to make a copy of your MS-DOS disk which is in drive A:, type

DISKCOPY A: B:

MS-DOS responds:

Insert source diskette into drive A:

Insert target diskette into drive B:

Strike any key when ready

Make sure the MS-DOS disk is in drive A: and insert a blank, *formatted* disk in drive B:. Press any character key after you have done this and MS-DOS will begin copying the MS-DOS disk. After MS-DOS has copied the disk, MS-DOS displays:

```
Copy complete

Copy another (Y/N)?
```

Type Y (for Yes) if you wish to copy another disk with DISKCOPY. If you type N (for No), the default drive prompt is displayed.

You now have a duplicate copy of your MS-DOS disk in drive B:. This duplicate copy can be saved as your backup copy of the MS-DOS disk.

Figure 2 illustrates the DISKCOPY command:

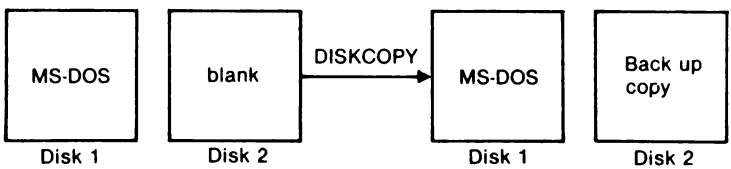


Figure 2. The DISKCOPY Command

Disks must be the same size and density to be copied with the DISKCOPY command. Refer to Chapter 5,"MS-DOS Commands", for more information on the DISKCOPY command.

NOTE

If either of the disks that you are using has defective tracks, DISKCOPY will not work. Use the COPY command to back up your disks in these cases. (COPY will skip over defective tracks.) Refer to Chapter 5, "MS-DOS Commands", for information on how to use COPY to back up your disks.

2.6 AUTOMATIC PROGRAM EXECUTION

If you want to run a specific program automatically each time you start MS-DOS, you can do so with Automatic Program Execution. For example, you may want to have MS-DOS display the names of your files each time you load MS-DOS.

When you start MS-DOS, the command processor searches for a file named AUTOEXEC.BAT on the MS-DOS disk. This file is a program that MS-DOS will run each time MS-DOS is started. Chapter 4, "Learning About Commands", tells you how to create an AUTOEXEC.BAT file.

2.7 FILES

2.7.1 What Is A File?

A file is a collection of related information. A file on your disk can be compared to a file folder in a desk drawer. For example, one file folder might contain the names and addresses of the employees who work in the office. You might name this file the Employee Master File. A file on your disk could also contain the names and addresses of employees in the office and could be named Employee Master File.

All programs, text, and data on your disk reside in files and each file has a unique name. You refer to files by their names. Chapter 3, "More About Files", tells you how to name your files.

You create a file each time you enter and save data or text at your terminal. Files are also created when you write and name programs and save them on your disks.

2.7.2 How MS-DOS Keeps Track Of Your Files

The names of files are kept in directories on a disk. These directories also contain information on the size of the files, their location on the disk, and the dates that they were created and updated. The directory you are working in is called your current or *working* directory.

An additional system area is called the File Allocation Table. It keeps track of the location of your files on the disk. It also allocates the free space on your disks so that you can create new files.

These two system areas, the directories and the File Allocation Table, enable MS-DOS to recognize and organize the files on your disks. The File Allocation Table is copied onto a new disk when you format it with the MS-DOS FORMAT command and one empty directory is created, called the *root* directory.

2.7.3 The DIR (Show Directory) Command

If you want to know what files are on your disk, you can use the DIR command. This command tells MS-DOS to display all the files in the current directory on the disk that is named. For example, if your MS-DOS disk is in drive A: and you want to see the listing for the current directory on that disk, type:

```
DIR A:
```

MS-DOS will respond with a directory listing of all the files in the current directory on your MS-DOS disk. The display should look similar to this:

```
 Volume in drive A has no label
 Directory of A: \

ANSI     SYS      1794     10-1-85   12:00p
ASSIGN   COM      1509     10-1-85   12:00p
ATTRIB   EXE     15091     10-1-85   12:00p
BACKUP   EXE     21720     10-1-85   12:00p
BASIC    COM      2048     10-1-85   12:00p
BASICA   COM      2048     10-1-85   12:00p
BASICA   EXE     68448     10-1-85   12:00p
CHKDSK   COM      9435     10-1-85   12:00p
COMMAND  COM     23210     10-1-85   12:00p
COMP     COM      2371     10-1-85   12:00p
DEBUG    COM     15552     10-1-85   12:00p
DISKCOMP COM      2352     10-1-85   12:00p
DISKCOPY COM      2643     10-1-85   12:00p
EDLIN    COM      7261     10-1-85   12:00p
EXE2BIN  EXE      2816     10-1-85   12:00p
FDISK    COM      6336     10-1-85   12:00p
FIND     EXE      6403     10-1-85   12:00p
FORMAT   COM      7947     10-1-85   12:00p
GRAPHICS COM       629     10-1-85   12:00p
JOIN     EXE     15971     10-1-85   12:00p
LABEL    EXE      2750     10-1-85   12:00p
LINK     EXE     38422     10-1-85   12:00p
MODE     COM      2112     10-1-85   12:00p
MORE     COM       282     10-1-85   12:00p
PARK     COM       281     10-1-85   12:00p
PRINT    COM      8291     10-1-85   12:00p
RECOVER  COM      4050     10-1-85   12:00p
RESTORE  EXE     20156     10-1-85   12:00p
SHARE    EXE      8304     10-1-85   12:00p
SORT     EXE      1664     10-1-85   12:00p
SUBST    EXE     16611     10-1-85   12:00p
SYS      COM      3727     10-1-85   12:00p
TREE     COM      1319     10-1-85   12:00p
VDISK    SYS      4301     10-1-85   12:00p
        32 File(s)        129024 bytes free
```

NOTE

Two MS-DOS system files, IO.SYS and MSDOS.SYS, are "hidden" files and will not appear when you issue the DIR command.

You can also get information about any file on your disk by typing DIR and a filename. For example, if you have created a file named MYFILE.TXT, the command

```
DIR MYFILE.TXT
```

will give you a display of all the directory information (name of file, size of file, date last edited) for the file MYFILE.TXT.

For more information on the DIR command, refer to Chapter 5, "MS-DOS Commands"

2.7.4 The CHKDSK (Check disk) Command

The MS-DOS command CHKDSK is used to check your disks for consistency and errors, much like a secretary proof reading a letter. CHKDSK analyzes the directories and the File Allocation Table on the disk that you specify. It then produces a status report of any inconsistencies, such as files which have a non-zero size in their directory but really have no data in them.

To check the disk in drive A:,type:

```
CHKDSK A:
```

MS-DOS will display a status report and any errors that it has found. An example of this display and more information on CHKDSK can be found in the description of the CHKDSK command in Chapter 5. You should run CHKDSK occasionally for each disk to ensure the integrity of your files.

2.8 HOW TO TURN THE SYSTEM OFF

There is no "logoff" command in MS-DOS. To end your terminal session, remove the disks. Then simply turn your computer off in response to a default drive prompt. Refer to the Owner's Manual for information on how to do this.

NOTE

Always remove your disks from the disk drives before you turn off your computer.

Summary of Commands in This Chapter

COMMAND	PURPOSE	SYNTAX
FORMAT	Formats disks for MS-DOS	FORMAT [*d:*]
DISKCOPY	Copies disks	DISKCOPY [*drive1:*] [*drive2:*]
DIR	Lists directory information	DIR [*d:*] [*filename*]
CHKDSK	Checks for errors on disk	CHKDSK [*d:*]

In the next chapter, you will learn more about MS-DOS files.

CHAPTER 3

MORE ABOUT FILES

How To Name Your Files
Wild Cards
  The ? Wild Card
  The * Wild Card
Illegal Filenames
How To Copy Your Files
How To Protect Your Files
Directories
Filenames And Paths
  Pathnames
  Pathing And External Commands
  Pathing And Internal Commands
  Displaying Your Working Directory
  Creating A Directory
  How To Change Your Working Directory
  How To Remove A Directory

In Chapter 2, you learned that directories contain the names of your files. In this chapter, you will learn how to name and copy your files. You will also learn more about the MS-DOS hierarchical directory structure which makes it easy for you to organize and locate your files.

3.1 HOW TO NAME YOUR FILES

The name of a typical MS-DOS file will look like this:

```
NEWFILE.EXE
```

The name of a file consists of two parts. The *filename* is NEWFILE and the *filename extension* is .EXE.

A filename can be from 1 to 8 characters long. The filename extension can be three or fewer characters. You can type any filename in small or capital letters and MS-DOS will translate these letters into uppercase characters.

In addition to the filename and the filename extension, the name of your file may include a *drive designation*. A drive designation tells MS-DOS to look on the disk in the designated drive to find the filename typed. For example, to find directory information about the file NEWFILE.EXE which is located on the disk in drive A: (and drive A: is NOT the default drive), type the following command:

```
DIR A:NEWFILE.EXE
```

Directory information about the file NEWFILE.EXE is now displayed on your screen.

If drive A: is the default drive, MS-DOS will search only the disk in drive A: for the filename NEWFILE and so the drive designation is not necessary. A drive designation is needed if you want to tell MS-DOS to look on the other drive to find a file.

Your filenames will probably be made up of letters and numbers, but other characters are allowed, too. Legal characters for filename extensions are the same as those for filenames. Here is a complete list of the characters you can use in filenames and extensions:

A-Z 0-9 $ & #

% . ' ( ) - @

^ { } ~ ` !

All of the parts of a filename comprise a file specification. The term file specification (or filespec) will be used in this manual to indicate the following filename format:

[<*drive designation:*>]<*filename*>
[<*.filename extension*>]

Remember that brackets indicate optional items. Angle brackets (< >) mean that you supply the text for the item. Note that the drive designation is not required unless you need to indicate to MS-DOS on which disk to search for a specific file. You do not have to give filename a filename extension.

Examples of file specifications are:

```
B:MYPROG.COB
A:YOURPROG.EXT
A:NEWFILE.
TEXT
```

3.2 WILD CARDS

Two special characters (called wild cards) can be used in filenames and extensions: the asterisk (*) and the question mark (?). These special characters give you greater flexibility when using filenames in MS-DOS commands.

3.2.1 The ? Wild Card

A question mark (?) in a filename or filename extension indicates that any character can occupy that position. For example, the MS-DOS command

```
DIR TEST?RUN.EXE
```

will list all directory entries on the default drive that have 8 characters, begin with TEST, have any next character, end with the letters RUN, and have a filename extension of .EXE. Here are some examples of files that might be listed by the above DIR command:

```
TEST1RUN.EXE
TEST2RUN.EXE
TEST6RUN.EXE
```

3.2.2 The * Wild Card

An asterisk (*) in a filename or filename extension indicates that any character can occupy that position *or any of the remaining positions in the filename or extension.*
For example:

```
DIR TEST*.EXE
```

will list all directory entries on the default drive with filenames that begin with the characters TEST and have an extension of EXE. Here are some examples of files that might be listed by the above DIR command:

```
TEST1RUN.EXE
TEST2RUN.EXE
TEST6RUN.EXE
TESTALL.EXE
```

The wild card designation *.* refers to *all* files on the disk. Note that this can be very powerful and destructive when used in MS-DOS commands. For example, the command DEL *.* deletes all files on the default drive, regardless of filename or extension.

Examples:

To list the directory entries for all files named NEWFILE on drive A: (regardless of their filename extensions), simply type:

```
DIR A:NEWFILE.*
```

To list the directory entries for all files with filename extensions of.TXT (regardless of their filenames) on the disk in drive B:, type:

```
DIR B:????????.TXT
```

This command is useful if, for example, you have given all your text programs a filename extension of .TXT. By using the DIR command with the wild card characters, you can obtain a listing of all your text files even if you do not remember all of their filenames.

3.3 ILLEGAL FILENAMES

MS-DOS treats some device names specially, and certain names are reserved for the names of these devices. These names cannot be used as filenames or extensions. You must not name your files any of the following:

CLOCK$

COM1,COM2,AUX
Used when referring to input from or output to an auxiliary device (such as a printer or disk drive).

CON Used when referring to keyboard input or to output to the terminal console (screen).

LPT1, LPT2, LPT3 or
PRN Used when referring to the printer device.

NUL Used when you do not want to create a particular file, but the command requires an input or output filename.

Even if you add device designations or filename extensions to these filenames, they remain associated with the devices listed above. For example, A:CON.XXX still refers to the console and is not the name of a disk file.

3.4 HOW TO COPY YOUR FILES

Just as with paper files, you often need more than one copy of a disk file. The COPY command allows you to copy one or more files to another disk. You can also give the copy a different name if you specify the new name in the COPY command.

The COPY command can also make copies of files on the same disk. In this case, you *must* supply MS-DOS with a different filename or you will overwrite the file. You cannot make a copy of a file on the same disk unless you specify a different filename for the new copy.

The format of the COPY command is:

COPY *filespec* [*filespec*]

For example,

COPY A:MYFILE.TXT B:MYFILE.TXT

will copy the file MYFILE.TXT on the disk in drive A: to a file named MYFILE.TXT on the disk in drive B:. A duplicate copy of MYFILE.TXT now exists.

Figure 3 illustrates how to copy files to another disk:

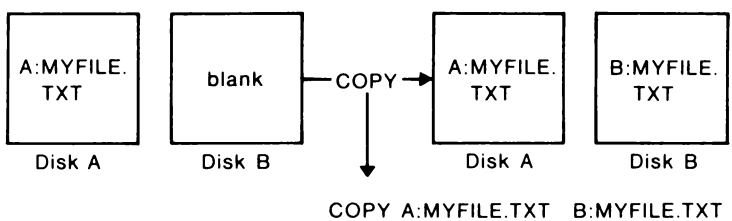


Figure 3. Copying Files to Another Disk

If you want to duplicate the file named MYFILE.TXT *on the same disk,* type:

COPY A:MYFILE.TXT A:NEWNAME.TXT

You now have two copies of your file on disk A—one named MYFILE.TXT and the other named NEWNAME.TXT. The following figure illustrates this example.

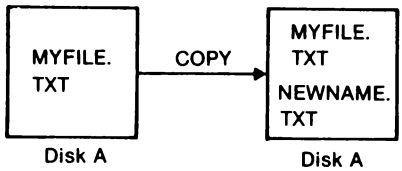


Figure 4. Copying Files on the Same Disk

You can also copy all files on a disk to another disk (i.e.make a backup copy) with the COPY command. Refer to Chapter 5, "MS-DOS Commands", for more information on this process.

3.5 HOW TO PROTECT YOUR FILES

MS-DOS is a powerful and useful tool in processing your personal and business information. As with any information system, inadvertent errors may occur and information may be misused. If you are processing information that cannot be replaced or requires a high level of security, you should take steps to ensure that your data and programs are protected from accidental or unauthorized use, modification, or destruction. Simple measures you can take—such as removing your disks when they are not in use, keeping backup copies of valuable information, and installing your equipment in a secure facility— can help you maintain the integrity of the information in your files.

3.6 DIRECTORIES

As you learned in Chapter 2, the names of your files are kept in a directory on each disk. The directory also contains information on the size of the files, their locations on the disk, and the dates when they were created and updated.

When there are multiple users on your computer, or when you are working on several different projects, the number of files in the directory can become large and unwieldy. You may want your own files kept separate from a co-worker's; or, you may want to organize your programs into categories that are convenient for you.

In an office, you can separate files by putting them in different filling cabinets; in effect, creating different directories of information. MS-DOS allows you to organize the files on your disks into directories. Directories are a way of dividing your files into convenient groups of files. For example, you may want all of your accounting programs in one directory and text files in another. Any one directory can contain any reasonable number of files, and it may also contain other directories (referred to as subdirectories).This method of organizing your files is called a hierarchical directory structure.

A hierarchical directory structure can be thought of as a "tree" structure: directories are branches of the tree and files are the leaves, except that the "tree" grows downward; that is, the "root" is at the top. The root is the first level in the directory structure. It is the directory that is automatically created when you format a disk and start putting files in it. You can create additional directories and subdirectories by following the instructions in Chapter 4, "Learning About Commands".

The tree or file structure grows as you create new directories for groups of files or for other people on the system. Within each new directory, files can be added, or new subdirectories can be created.

It is possible for you to "travel" around this tree; for instance, it is possible to find any file in the system by starting at the root and traveling down any of the branches to the desired file. Conversely, you can start where you are within the file system and travel towards the root.

The filenames discussed earlier in this chapter are relative to your current directory and do not apply system-wide. Thus, when you turn on your computer, you are "in" your directory. Unless you take special action when you create a file, the new file is created in the directory in which you are now working. Users can have files of the same name that are unrelated because each is in a different directory.

Figure 5 illustrates a typical hierarchical directory structure.

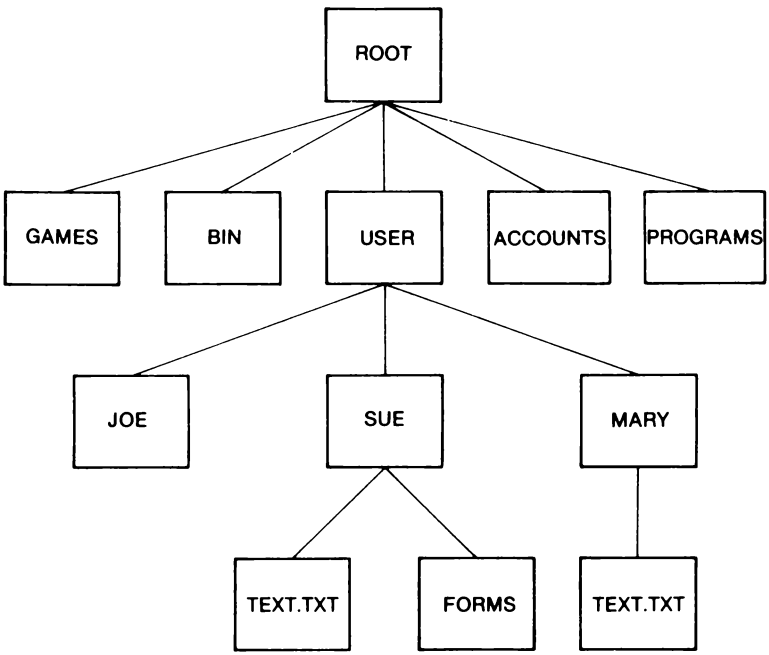


Figure 5. A Sample Hierarchical Directory Structure

The ROOT directory is the first level in the directory structure. You can create subdirectories from the ROOT by using the MKDIR command (refer to Chapter 5, "MS-DOS Commands", for information on MKDIR). In this example, five subdirectories of ROOT have been created. These include:

A directory of games, named GAMES

A directory of all external commands, named BIN (refer to Chapter 4, "Learning About Commands", for more information on the BIN directory)

A USER directory containing separate subdirectories for all users of the system

A directory containing accounting information, named ACCOUNTS

A directory of programs, named PROGRAMS

Joe, Sue, and Mary each have their own directories which are subdirectories of the USER directory. Sue has a subdirectory under the \ USER \ SUE directory named FORMS. Sue and Mary have files in their directories, each named TEXT.TXT. Notice that Mary's text file is unrelated to Sue's.

This organization of files and directories is not important if you only work with files in your own directory; but if you work with someone else or on several projects at one time, the hierarchical directory structure becomes extremely useful. For example, you could get a list of the files in Sue's FORMS directory by typing:

```
DIR\USER\SUE\FORMS
```

Note that the backward slash mark (\ ) is used to separate directories from other directories and files.

To find out what files Mary has in her directory, you could type:

```
DIR\USER\MARY
```

3.7 FILENAMES AND PATHS

When you use hierarchical directories, you must tell MS-DOS where the files are located in the directory structure. Both Mary and Sue, for example, have files named TEXT.TXT. Each will have to tell MS-DOS in which directory her file resides if she wants to access it. This is done by giving MS-DOS a *pathname* to the file.

3.7.1 Pathnames

A simple filename is a sequence of characters that can optionally be preceded by a drive designation and followed by an extension. A pathname is a sequence of directory names followed by a simple filename, each separated from the previous one by a slash(\ ).

The syntax of pathnames is:

[<*d*>:][<*directory*>]\[<*directory...*>]\
[<*filename*>]

If pathname begins with a slash, MS-DOS searches for the file beginning at the root (or top) of the tree. Otherwise, MS-DOS begins at the user's current directory, known as the working directory, and searches downward from there. The pathname of Sue's TEXT.TXT file is \ USER \ SUE \ TEXT.TXT.

When you are in your working directory, a filename and its corresponding pathname may be used interchangeably. Some sample names are:

\ Indicates the root directory.

\ PROGRAMS Sample directory under the root directory containing program files.

\ USER \ MARY \ FORMS \ 1A
A typical full pathname. This one happens to be a file named 1A in the directory named FORMS belonging to the USER named MARY.

USER \ SUE A relative pathname; it names the file or directory SUE in subdirectory USER of the working directory. If the working directory is the root (\ ), (\ ), it names \ USER \ SUE.

TEXT.TXT Name of a file or directory in the working directory.

MS-DOS provides special shorthand notations for the *working* directory and the *parent* directory (one level up) of the working directory:

MS-DOS uses this shorthand notation to indicate the name of the working directory in all hierarchical directory listings.
MS-DOS automatically creates this entry when a directory is made.

The shorthand name of the working directory's parent directory.

If you type:

DIR..

then MS-DOS will list the files in the parent directory of your working directory.

If you type:

DIR.. \..

then MS-DOS will list the files in the parent's PARENT directory.

3.7.2 Pathing And External Commands

External commands reside on disks as program files. They must be read from the disk before they execute. (For more information on external commands, refer to Chapter 4, "Learning About Commands".)

When you are working with more than one directory, it is convenient to put all MS-DOS external commands into a separate directory so they do not clutter your other directoies. When you issue an external command to MS-DOS, MS-DOS immediately checks your working directory to find that command. You must tell MS-DOS in which directory these external commands reside. This is done with the PATH command.

For example, if you are in a working directory named \ BIN \ PROG, and all MS-DOS external commands are in \ BIN, you must tell MS-DOS to choose the \ BIN path to find the FORMAT command. The command

```
PATH \BIN
```

tells MS-DOS to search in your working directory and the \ BIN directory for all commands. You only have to specify this path once to MS-DOS during your terminal session. MS-DOS will now search in \ BIN for the external commands. If you want to know what the current path is, type the word PATH and the current value of PATH will be printed.

For more information on the MS-DOS command PATH, refer to Chapter 5, "MS-DOS Commands".

3.7.3 Pathing And Internal Commands

Internal commands are the simplest, most commonly used commands. They execute immediately because they are incorporated into the command processor. (For more information on internal commands, refer to Chapter 4, "Learning About Commands".)

Some internal commands can use paths. The following four commands, COPY, DIR, DEL, and TYPE have greater flexibility when you specify a pathname after the command.

The syntax of these four commands is shown below.

COPY <*pathname pathname*>
: If the second pathname to COPY is a directory, all files are copied into that directory.

DEL <*pathname*>
: If the pathname is a directory, all the files in that directory are deleted. Note: The Prompt "Are you sure (Y/N) ?" will be displayed if you try to delete a path. Type Y to complete the command, or type N for the command to abort.

DIR <*pathname*>
: Displays the directory for a specific path.

TYPE <*pathname*>

You must specify a file in a path for this command. MS-DOS will display the file on your screen in response to the TYPE pathname command.

3.7.4 Displaying Your Working Directory

All commands are executed while you are in your working directory. You can find out the name of the directory you are in by issuing the MS-DOS command CHDIR (Change Directory) with no options. For example, if your current directory is \ USER \ JOE, when you type:

CHDIR<ENTER>

You will see:

A:\USER\JOE

This is your current drive designation plus the working directory (\ USER \ JOE).

If you now want to see what is in the \ USER \ JOE directory, you can issue the MS-DOS command DIR. The following is an example of the display you might receive from the DIR command for a subdirectory:

```
Volume in drive A has no ID
Directory of A:\USER\JOE

                <DIR>          8-09-82  10:09a
                <DIR>          8-09-82  10:09a
TEXT            <DIR>          8-09-82  10:09a
FILE1    COM             5243  8-04-82   9:30a
     4 File(s)     298320 bytes free
```

A volume ID for this disk was not assigned when the disk was formatted. Note that MS-DOS lists both files and directories in this output. As you can see, Joe has another directory in this tree structure named TEXT. The '.' indicates the working directory \ USER \ JOE, and the '..' is the shorthand notation for the parent directory \ USER. FILE1.COM is a file in the \ USER \ JOE directory. All of these directories and files reside on the disk in drive A:.

Because files and directories are listed together (see previous display), MS-DOS does not allow you to give a subdirectory the same name as a file in that directory. For example, if you have a path \BIN\USER\JOE where JOE is a subdirectory, you cannot create a file in the USER directory named JOE.

3.7.5 Creating A Directory

To create a subdirectory in your working directory, use the MKDIR (Make Directory) command. For example, to create a new directory named NEWDIR under your working directory, simply type:

```
A>MKDIR NEWDIR
```

After this command has been executed by MS-DOS, a new directory will exist in your tree structure under your working directory. You can also make directories anywhere in the tree structure by specifying MKDIR and then a pathname. MS-DOS will automatically create the and .. entries in the new directory.

There is a limit on the length of the directory pathname:

```
a:\attorney\jacobson\contracts\ellis.new
  └──────────────────────────────────────┘

from here <----------------------------------------------------------------------> to here
```

This pathname cannot be over 63 characters long.

To put files in the new directory, use the MS-DOS line editor. EDLIN. Chapter 7, "The Line Editor (EDLIN)", describes how to use EDLIN to create and save files.

3.7.6 How To Change Your Working Directory

Changing from your working directory to another directory is very easy in MS-DOS. Simply issue the CHDIR (Change Directory) command and supply a pathname. For example:

```
A>CHDIR \USER
```

changes the working directory from \ USER \ JOE to \ USER. You can specify any pathname after the command to "travel" to different branches and leaves of the directory tree. The command "CHDIR.."will always put you in the parent directory of your working directory.

3.7.7 How To Remove A Directory

To delete a directory in the tree structure, use the MS-DOS RMDIR (Remove Directory) command. For example, to remove the directory NEWDIR from the working directory, type:

```
RMDIR NEWDIR
```

Note that the directory NEWDIR must be *empty* except for the . and .. entries before it can be removed; this will prevent you from accidentally deleting files and directories. You can remove any directory by specifying its pathname. To remove the \BIN\USER\JOE directory, make sure that it has only the . and .. entries, then type:

```
RMDIR \BIN\USER\JOE
```

To remove all the *files* in a directory (except for the . and .. entries), type DEL and then the pathname of the directory. For example, to delete all files in the \ BIN \ USER \ SUE directory, type:

```
DEL\BIN\USER\SUE
```

You cannot delete the . and .. entries. They are treated by MS-DOS as part of the hierarchical directory structure.

Summary of Commands in This Chapter

COMMAND	PURPOSE	SYNTAX
COPY	Copies files	COPY filespec [*filespec*]
PATH	Sets MS-DOS search path	PATH [*pathname*]
CHDIR	Displays working directory; changes directories	CHDIR [*pathname*]
MKDIR	Makes a new directory	MKDIR [*pathname*]
RMDIR	Removes a directory	RMDIR [*pathname*]

In the next chapter, you will learn about MS-DOS commands.

CHAPTER 4

LEARNING ABOUT COMMANDS

Introduction
Types Of MS-DOS Commands
Command Options
Information Common To All MS-DOS Commands
Batch Processing
The AUTOEXEC.BAT File
  How To Create An AUTOEXEC.BAT File
Creating A .BAT File With Replaceable Parameters
  Executing A .BAT File
Input And Output
  Redirecting Your Output
  Filters
  Command Piping

4.1 INTRODUCTION

Commands are a way of communicating with the computer. By entering MS-DOS commands at your terminal, you can ask the system to perform useful tasks. There are MS-DOS commands that:

Compare, copy, display, delete, and rename files

Copy and format disks

Execute system programs such as EDLIN, as well as your own programs

Analyze and list directories

Enter date, time, and remarks

Set various printer and screen options

Copy MS-DOS system files to another disk

Request MS-DOS to wait for a specific period of time

4.2 TYPES OF MS-DOS COMMANDS

There are two types of MS-DOS commands:

Internal commands

External commands

Internal commands are the simplest, most commonly used commands. You cannot see these commands when you do a directory listing on your MS-DOS disk; they are part of the command processor. When you type these commands, they execute immediately. The following internal commands are described in Chapter 5:

BREAK	DEL (ERASE)	MKDIR (MD)	SET
CHDIR (CD)	DIR	PATH	SHIFT
CLS	ECHO	PAUSE	TIME
COPY	EXIT	PROMPT	TYPE
CTTY	FOR	REM	VER
DATE	GOTO	REN (RENAME)	VERIFY
	IF	RMDIR (RD)	VOL

External commands reside on disks as program files. They must be read from disk before they can execute. If the disk containing the command is not in the drive, MS-DOS will not be able to find and execute the command.

Any filename with a filename extension of .COM, .EXE or .BAT is considered an external command. For example, programs such as FORMAT.COM and COMP.COM are external commands. Because all external commands reside on disk, you can create commands and add them to the system. Programs that you create with most languages (including assemdly language) will be .EXE (executable) files.

When you enter an external command, do not include its filename extension. The following external commands are described in Chapter 5:

ASSIGN	FIND	PRINT
ATTRIB	FORMAT	RECOVER
BACKUP	GRAPHICS	RESTORE
CHKDSK	JOIN	SHARE
COMP	LABEL	SORT
DISKCOMP	MODE	SUBST
DISKCOPY	MORE	SYS
EXE2BIN	PARK	TREE

4.3 COMMAND OPTIONS

Options can be included in your MS-DOS commands to specify additional information to the system. If you do not include some options, MS-DOS provides a default value. Refer to individual command descriptions in Chapter 5 for the default values.

The following is the format of all MS-DOS commands:

Command [*options...*]

where:

d: Refers to disk drive designation.

filename Refers to any valid name for a disk file, including an optional filename extension. The filename option does not refer to a device or to a disk drive designation.

ext Refers to an optional filename extension consisting of a period and 1-3 characters. When used, filename extensions immediately follow filenames.

filespec Refers to an optional drive designation, a filename, and an optional three letter filename extension in the following format:

[<*d:*>]<*filename*>[<*.ext*>]

pathname Refers to a pathname or filename in the following format:

[<*directory*>]\[<*directory...*>]\
[<*filename*>]

switches Switches are options that control MS-DOS commands. They are preceded by a forward slash (for example, /P).

arguments Provide more information to MS-DOS commands. You usually choose between arguments; for example, ON or OFF.

4.4 INFORMATION COMMON TO ALL MS-DOS COMMANDS

The following information applies to all MS-DOS commands:

1. Commands are usually followed by one or more options.

2. Commands and options may be entered in uppercase or lowercase, or a combination of keys.

3. Commands and options must be separated by delimiters. Because they are easiest, you will usually use the space and comma as delimiters. For example:

   ```
   DEL MYFILE.OLD NEWFILE.TXT
   RENAME,THISFILE THATFILE
   ```

   You can also use the semicolon (;), the equal sign (=), or the tab key as delimiters in MS-DOS commands.

   In this manual, we will use a space as the delimiter in commands.

4. Do not separate a file specification with delimiters, since the colon and the period already serve as delimiters.

5. When instructions say "Press *any* key", you can press any alpha (A-Z) or numeric (0-9) key.

6. You must include the filename extension when referring to a file that already has a filename extension.

7. You can abort commands when they are running by pressing <CONTROL-C>

8. Commands take effect only after you have pressed the <ENTER> key.

9. Wild cards (global filename characters) and device names (for example, PRN or CON) are not allowed in the names of any commands.

10. When commands produce a large amount of output on the screen, the display will automatically scroll to the next screen. You can press <CONTROL-S> to suspend the display. Press any key to resume the display on the screen.

11. MS-DOS editing and function keys can be used when entering commands. Refer to Chapter 6, "MS-DOS Editing and Function Keys", for a complete description of these keys.

12. The prompt from the command processor is the default drive designation plus a greater-than sign; for example,A>.

13. Disk drives will be referred to as *source* drives and *destination* drives. A source drive is the drive you will be transferring information from. A destination drive is the drive you will be transferring information to.

4.5 BATCH PROCESSING

Often you may find yourself typing the same sequence of commands over and over to perform some commonly used task. With MS-DOS, you can put the command sequence into a special file called a batch file, and execute the entire sequence simply by typing the name of the batch file. "Batches" of your commands in such files are processed as if they were typed at a terminal. Each batch file must be named with the .BAT extension, and is executed by typing the filename without its extension.

You can create a batch file by using the Line Editor (EDLIN) or by typing the COPY command. Refer to the "How to Create an AUTOEXEC.BAT File" section later in this chapter for more information on using the COPY command to create a batch file.

Two MS-DOS commands are available for use expressly in batch files: REM and PAUSE. REM permits you to include remarks and comments in your batch files without these remarks being executed as commands. PAUSE prompts you with an optional message and permits you to either continue or abort the batch process at a given point. REM and PAUSE are described in detail in Chapter 5.

Batch processing is useful if you want to execute several MS-DOS commands with one batch command, such as when you format and check a new disk. For example, a batch file for this purpose might look like this:

```
1:  REM This is a file to check new disks
2:  REM It is named NEWDISK.BAT
3:  PAUSE Insert new disk in drive B:
4:  FORMAT B:
5:  DIR B:
6:  CHKDSK B:
```

To execute this .BAT file, simply type the filename without the BAT extension:

```
NEWDISK
```

The result is the same as if each of the lines in the .BAT file was entered at the terminal as individual commands.

Figure 6 illustrates the three steps used to write, save, and execute an MS-DOS batch file.

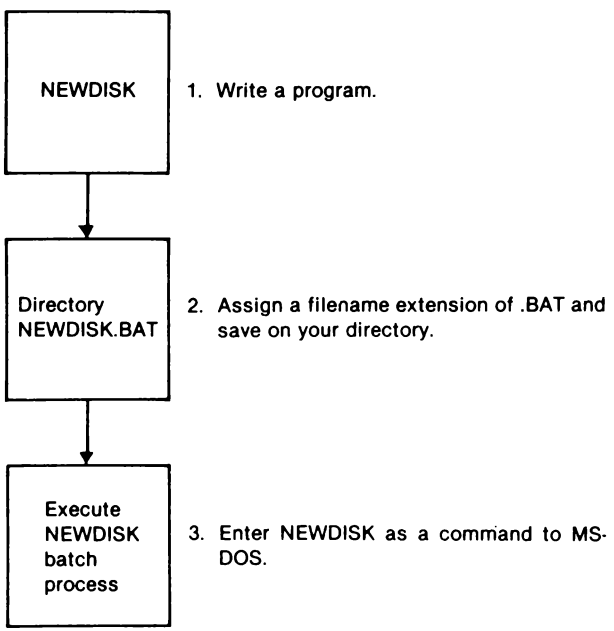


Figure 6. MS-DOS Batch File Steps

The following list contains information that you should read before you execute a batch process with MS-DOS:

1. Do not enter the filename BATCH (unless the name of the file you want to execute is BATCH.BAT).

2. Only the filename should be entered to execute the batch file. Do not enter the filename extension.

3. The commands in the file named <*filename*>. BAT are executed.

4. If you press <CONTROL-C> while in batch mode, this prompt appears:

   Terminate batch job (Y/N)?

   If you press Y, the remainder of the commands in the batch file are ignored and the system prompt appears.

   If you press N, only the current command ends and batch processing continues with the next command in the file.

5. If you remove the disk containing a batch file being executed, MS-DOS prompts you to insert it again before the next command can be read.

6. The last command in a batch file may be the name of another batch file. This allows you to call one batch file from another when the first is finished.

4.6 THE AUTOEXEC.BAT FILE

As discussed in Chapter 2, an AUTOEXEC.BAT file allows you to automatically execute programs when you start MS-DOS. Automatic Program Execution is useful when you want to run a specific package (for example, Microsoft Multiplan) under MS-DOS, and when you want MS-DOS to execute a batch program automatically each time you start the system. You can avoid loading two separate disks to perform either of these tasks by using an AUTOEXEC.BAT file.

When you start MS-DOS, the command processor searches the MS-DOS disk for a file named AUTOEXEC.BAT. The AUTOEXEC.BAT file is a batch file that is automatically executed each time you start the system.

If MS-DOS finds the AUTOEXEC.BAT file, the file is immediately executed by the command processor and the date and time prompts are bypassed.

If MS-DOS does not find an AUTOEXEC.BAT file when you first load the MS-DOS disk, then the date and time prompts will be issued. Figure 7 illustrates how MS-DOS uses the AUTOEXEC.BAT file.

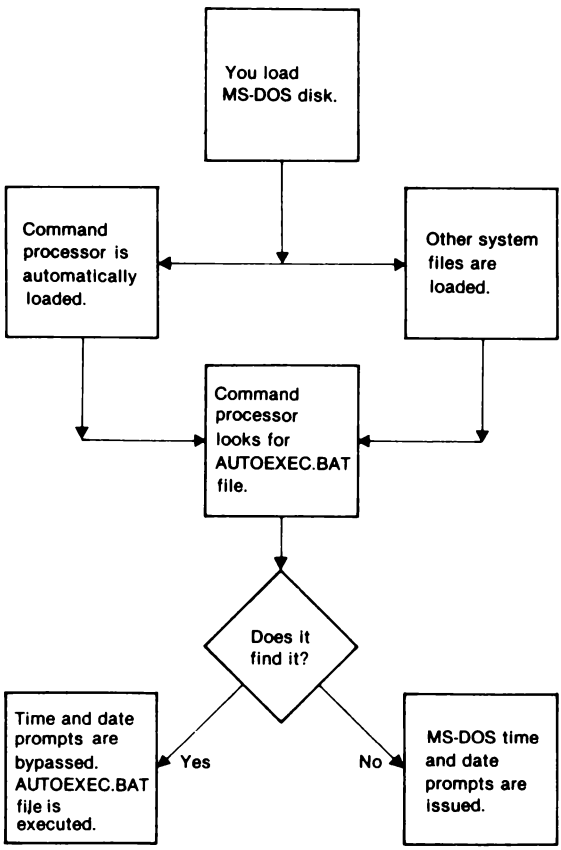

You load MS-DOS disk.
Command processor is automatically loaded.
Other system files are loaded.
Command processor looks for AUTOEXEC.BAT file.
Does it find it?
Yes
No
Time and date prompts are bypassed. AUTOEXEC.BAT file is executed.
MS-DOS time and date prompts are issued.

Figure 7. How MS-DOS Uses the AUTOEXEC.BAT File

4.6.1 How To Create An AUTOEXEC.BAT File

If, for example, you wanted to automatically load BASIC and run a program called MENU each time you started MS-DOS, you could create an AUTOEXEC.BAT file as follows:

1. Type:

   COPY CON: AUTOEXEC.BAT

   This statement tells MS-DOS to copy the information from the console (keyboard) into the AUTOEXEC.BAT file. Note that the AUTOEXEC.BAT file *must* be created in the root directory of your MS-DOS disk.

2. Now type:

   BASICA MENU

   This statement goes into the AUTOEXEC.BAT file. It tells MS-DOS to load BASIC and run the MENU program whenever MS-DOS is started.

3. Press the <CONTROL-Z> key; then press the <ENTER> key to put the command BASICA MENU in the AUTOEXEC.BAT file.

4. The MENU program will now run automatically whenever you start MS-DOS.

To run your own BASIC program, enter the name of your program in place of MENU in the second line of the example. You can enter any MS-DOS command or series of commands in the AUTOEXEC.BAT file.

NOTE

Remember that if you use an AUTOEXEC.BAT file, MS-DOS will not prompt you for a current date and time unless you include the DATE and TIME commands in the AUTOEXEC.BAT file. It is strongly recommended that you include these two commands in your AUTOEXEC.BAT file, since MS-DOS uses this information to keep your directory current.

4.7 CREATING A .BAT FILE WITH REPLACEABLE PARAMETERS

There may be times when you want to create an application program and run it with different sets of data. These data may be stored in various MS-DOS files.

When used in MS-DOS commands, a parameter is an option that you define. With MS-DOS, you can create a batch (.BAT) file with dummy (replaceable) parameters. These parameters,named %0- %9, can be replaced by values supplied when the batch file executes.

For example, when you type the command line COPY CON MYFILE.BAT, the next lines you type are copied from the console to a file named MYFILE.BAT on the default drive:

```
A>COPY CON: MYFILE.BAT
COPY %1.MAC %2.MAC
TYPE %2.PRN
TYPE %0.BAT
```

Now, press <CONTROL-Z> and then press <ENTER>. MS-DOS responds with this message:

```
1 File (s) copied
A>_
```

The file MYFILE.BAT, which consists of three commands, now resides on the disk in the default drive.

The dummy parameters %1 and %2 are replaced sequentially by the parameters you supply when you execute the file. The dummy parameter %0 is always replaced by the drive designator, if specified, and the filename of the batch file (for example, MYFILE).

NOTES:

1. Up to 10 dummy parameters (%0-%9) can be specified. Refer to the MS-DOS command SHIFT in Chapter 5 if you wish to specify more than 10 parameters.

2. If you use the percent sign as part of a filename *within a batch file*, you must type it twice. For example, to specify the file ABC%.EXE, you must type it as ABC%%.EXE in the batch file.

4.7.1 Executing A .BAT File

To execute the batch file MYFILE.BAT and to specify the parameters that will replace the dummy parameters, you must enter the batch filename (without its extension) followed by the parameters you want MS-DOS to substitute for %1,%2, etc.

Remember that the file MYFILE.BAT consists of 3 lines:

```
COPY %1.MAC %2.MAC
TYPE %2.PRN
TYPE %0.BAT
```

To execute the MYFILE batch process, type:

```
MYFILE A:PROG1 B:PROG2
```

MYFILE is substituted for %0, A:PROG1 for %1, and B:PROG2 for %2.

The result is the same as if you had typed each of the commands in MYFILE with their parameters, as follows:

```
COPY A:PROG1.MAC B:PROG2.MAC
TYPE B:PROG2.PRN
TYPE MYFILE.BAT
```

The following table illustrates how MS-DOS replaces each of the above parameters:

BATCH FILENAME	PARAMETER1(%0) (MYFILE)	PARAMETER2(%1) (PROG1)	PARAMETER3(%2) (PROG2)
MYFILE	MYFILE.BAT	PROG1.MAC	PROG2.MAC PROG2.PRN

Remember that the dummy parameter %0 is always replaced by the drive designator (if specified) and the filename of the batch file.

4.8 INPUT AND OUTPUT

MS-DOS always assumes that input comes from the keyboard and output goes to the terminal screen. However,the flow of command input and output can be redirected. Input can come from a file rather than a terminal keyboard, and output can go to a file or to a line printer instead of to the terminal. In addition, "Pipes" can be created that allow output from one command to become the input to another. Redirection and pipes are discussed in the next sections.

4.8.1 Redirecting Your Output

Most commands produce output that is sent to your terminal. You can send this information to a file by using a greater-than sign (>) in your command. For example, the command

```
DIR
```

displays a directory listing of the disk in the default drive *on the terminal screen*. The same command can send this output to a file named MYFILES by designating the output file on the command line:

```
DIR>MYFILES
```

If the file MYFILES does not already exist, MS-DOS creates it and stores your directory listing in it. If MYFILES already exists, MS-DOS overwrites what is in the file with the new data.

If you want to append your directory or a file to another file (instead of replacing the entire file), two greater-than signs(> >) can be used to tell MS-DOS to append the output of the command (such as a directory listing) to the end of a specified file. The command

```
DIR>>MYFILES
```

appends your directory listing to a currently existing file named MYFILES. If MYFILES does not exist, it is created.

It is often useful to have input for a command come from a file rather than from a terminal. This is possible in MS-DOS by using a less-than sign (<) in your command. For example, the command

```
SORT<NAMES>LIST1
```

sorts the file NAMES and sends the sorted output to a file named LIST1.

4.8.2 Filters

A filter is a command that reads your input, transforms it in some way, and then outputs it, usually to your terminal or to a file. In this way, the data is said to have been "filtered" by the program. Since filters can be put together in many different ways, a few filters can take the place of a large number of specific commands.

MS-DOS filters include FIND, MORE, and SORT. Their functions are described below:

FIND Searches for a constant string of text in a file

MORE Takes standard terminal output and displays it, one screen at a time

SORT Sorts text

You can see how these filters are used in the next section.

4.8.3 Command Piping

If you want to give more than one command to the system at a time, you can "pipe" commands to MS-DOS. For example, you may occasionally need to have the output of one program sent as the input to another program. A typical case would be a program that produces output in columns. It could be desirable to have this columnar output sorted.

Piping is done by separating commands with the pipe separator, which is the vertical bar symbol (|). For example, the command

```
DIR | SORT
```

will give you an alphabetically sorted listing of your directory. The vertical bar causes all output generated by the left side of the bar to be sent to the right side of the bar for processing.

Piping can also be used when you want to output to a file. If you want your directory sorted and sent to a new file (for example, DIREC.FIL), you could type:

```
DIR | SORT > DIREC.FIL
```

MS-DOS will create a file named DIREC.FIL on your default drive. DIREC.FIL contains a sorted listing of the directory on the default drive, since no other drive was specified in the command. To specify a drive other than the default drive, type:

```
DIR | SORT > B:DIREC.FIL
```

This sends the sorted data to a file named DIREC.FIL on drive B:.

A pipeline may consist of more than two commands. For example.

```
DIR | SORT | MORE
```

will sort your directory, show it to you one screen at a time, and put — MORE — at the bottom of your screen when there is more output to be seen.

You will find many uses for piping commands and filters.

Summary of Commands in This Chapter

COMMAND	PURPOSE	SYNTAX
REM	Adds comment line for batch files	REM [*remark*]
PAUSE	Suspends execution of a batch file	PAUSE [*comment*]
FIND	Searches for string of text	FIND string [*filename*]
MORE	Pages through a file 23 lines at a time	MORE
SORT	Sorts text	SORT

You will find more information on using these filters in the next chapter, MS-DOS Commands.

CHAPTER 5

MS-DOS COMMANDS

Command Formats
Summary of MS-DOS Commands
Description of Commands
Batch Processing Commands

5.1 COMMAND FORMATS

The following notation indicates how you should format MS-DOS commands:

1. You must enter any words shown in capital letters. These words are called *keywords* and must be entered exactly as shown. You can enter these keywords in any combination of upper/lowercase; MS-DOS will convert all keywords to uppercase.

2. You supply the text for any items enclosed in angle brackets (< >). For example, you should enter the name of *your* file when <*filename*> is shown in the format.

3. Items in square brackets ([ ]) are optional. If you wish to include optional information, do not include the square brackets, only the information within the brackets.

4. An ellipsis (...) indicates that you may repeat an item as many times as you want.

5. You must include all punctuation where shown (with the exception of square brackets), such as commas, equal signs, question marks, colons, or slashes.

5.2 SUMMARY OF MS-DOS COMMANDS

NOTE

Users of single-drive systems should refer to Appendix A for the additional procedures required when executing many of the following commands.

The following MS-DOS commands are described in this chapter. Note that synonyms for commands are enclosed in parentheses.

ASSIGN	Assigns a different drive from the one that was currently assigned
ATTRIB	Sets or displays attributes of a file
BACKUP	Backs up files
BREAK	Sets CONTROL-C check
CHDIR	Changes directories; prints working directory (CD)
CHKDSK	Scans the directory of the default or designated drive and checks for consistency
CLS	Clears screen
COMP	Compares two files
COPY	Copies file(s) specified
CTTY	Changes console TTY
DATE	Displays and sets date
DEL	Deletes file(s) specified (ERASE)
DIR	Lists requested directory entries
DISKCOMP	Compares two entire disks
DISKCOPY	Copies disks
EXE2BIN	Converts executable files to binary format
EXIT	Exits command and returns to lower level
FIND	Searches for a constant string of text
FORMAT	Formats a disk to receive MS-DOS files
GRAPHICS	Prints contents of a screen on a printer
JOIN	Joins a disk drive to a pathname
LABEL	Gives a volume label to a disk
MKDIR	Makes a directory (MD)
MODE	Sets the mode of operation on peripherals
MORE	Displays output one screen at a time
PARK	Repositions read/write head on the hard disk
PATH	Sets a command search path
PRINT	Background print feature
PROMPT	Designates command prompt
RECOVER	Recovers a bad disk
REM	Displays a comment in a batch file
REN	Renames first file as second file(RENAME)
RESTORE	Restores backed up files
RMDIR	Removes a directory (RD)
SET	Sets one string value to another
SHARE	Installes file sharing and locking
SORT	Sorts data alphabetically, forward or backward
SUBST	Substitutes a string for a pathname
SYS	Transfers MS-DOS system files from drive A: to the drive specified

TIME	Displays and sets time
TREE	Displays all of the directory paths
TYPE	Displays the contents of file specified
VER	Prints MS-DOS version number
VERIFY	Verifies writes to disk
VOL	Prints volume identification number

Batch Commands (Command extensions)

ECHO	Turns batch file echo feature on/off
FOR	Batch command extension
GOTO	Batch command extension
IF	Batch command extension
PAUSE	Pauses for input in a batch file
SHIFT	Increases number of replaceable parameters in batch process

5.3 DESCRIPTION OF COMMANDS

ASSIGN

Name: ASSIGN

Type: External

Purpose: Assigns a drive letter to a different drive.

Syntax: [*drive:*] [<*pathname*>] ASSIGN [*x* = *y* [...]]

Comments: The ' = ' equal sign is optional.
x represents the current drive which accepts reads and writes.
y represents the new drive which you want to accept reads and writes.
Colons do not have to be typed after the drive letter.

The original assignment of all drives can be restored by typing:

```
ASSIGN
```

then press <ENTER>.

For programs that allow you to only perform disk operations on drives A and B, the ASSIGN command allows you to access other disk drives.
Use the command

```
ASSIGN A=C B=C
```

to redirect all input and output from drive A and B to drive C. This will enable you to use a hard disk (C) for such programs.

ATTRIB

Name: ATTRIB

Type: External

Purpose: Sets or resets the read-only attribute of a file, and displays the attributes of a file.

Syntax: [*drive*:] [*pathname*] ATTRIB [+R/−R] [*drive*:] [*pathname*]

Comments: Read-only is set by +R.
Read-only is disabled by −R.
The file is specified by [*drive*:] [*pathname*] that follow [+R/−R].
To permit file sharing over a network, the ATTRIB command forces a read-only mode even if the application software opens a file in a read and write mode.
The attributes of a file can be displayed by typing

ATTRIB <*pathname*>

then press <ENTER>.

The characters *.* can be used as wildcards to print the attributes of all files on a disk.
For example: To make the file named MYFILE.TXT read-only type:

ATTRIB +R MYFILE.TXT

BACKUP

Name: BACKUP

Type: External

Purpose: Backs up the files on a hard disk to a floppy disk.

Syntax: [*drive:*] [*pathname*] BACKUP [*drive:*] [*pathname*] [*drive:*] [/S] [/M] [/A] [/P] [/D:<*date*>] [/T:<*time*>] [/L:<*filename*>]

Comments: After the BACKUP command, the [*drive:*] specified is the hard disk drive containing file(s) to back up. The next [*drive:*] is the floppy disk drive used for the backup. The old files on the floppy disk are erased before the backup proceeds, unless otherwise specified with the /A switch.

File and disk formats are compatible with this program and the backup program supplied by IBM® , unless the /P or /T switch is set. IBM restore program may not restore files that are backed up with the /P switch.

BACKUP supports the following switches.

/S All subdirectories are backed up.

/M The files that have changed since the last backup are backed up.

/A Old files are not erased from the disk before new or current files are backed up.

/P Fill the floppy disk with as many files as possible by creating subdirectories if necessary. (CAUTION: The ability to use the IBM Backup /Restore program may be lost if this switch is used.)

/D The files at or after the specified date are backed up if they have been modified.

/T The files at or after the specified time are backed up if they have been modified.

/L This switch creates a backup log by the file name given or the file name BACKUP.LOG is created on the root directory if the file name was not specified. The contents of the file consists of the backup date and time on the first line. The following lines contain the file name and the floppy disk number. To restore a file from a floppy disk this information will save time in searching for a particular file. Current entries are appended if the backup log file already exists.

BREAK

Name: BREAK

Type: Internal

Purpose: Sets CONTROL-C check.

Syntax: BREAK [ON/OFF]

Comments: If you are running an application program that uses CONTROL-C function keys, you will want to turn off the MS-DOS CONTROL-C function so that when you press <CONTROL-C> you affect your program and not the operating system. Specify BREAK OFF to turn off CONTROL-C and BREAK ON when you have finished running your application program and are using MS-DOS.

CHDIR

Name: CHDIR (CHANGE DIRECTORY)

Type: Internal

Synonym: CD

Purpose: Changes directory to a different path; displays current (working) directory.

Syntax: CHDIR [*pathname*]

Comments: If your working directory is \ BIN \ USER \ JOE and you want to change your path to another directory (such as \ BIN \ USER \ JOE \ FORMS), type:

CHDIR \ BIN \ USER \ JOE \ FORMS

and MS-DOS will put you in the new directory. A shorthand notation is also available with this command:

CHDIR..

This command will always put you in the parent directory of your working directory.

CHDIR used without a pathname displays your working directory. If your working directory is \ BIN \ USER \ JOE on drive B:, and you type CHDIR <ENTER>, MS-DOS will display:

B: \ BIN \ USER \ JOE

This command is useful if you forget the name of your working directory.

CHKDSK

Name: CHKDSK (CHECK DISK)

Type: External

Purpose: Scans the directory of the specified disk drive and checks it for consistency.

Syntax: [*drive*:] [*pathname*] CHKDSK [*drive*:] <*filespec*> [/F] [/V]

Comments: CHKDSK should be run occasionally on each disk to check for errors in the directory. If any errors are found, CHKDSK will display error messages, if any, and then a status report.

A sample status report follows:

```
160256  bytes total disk space
  8192  bytes in 2 hidden files
   512  bytes in 2 directories
 30720  bytes in 8 user files
121344  bytes available on disk
 65536  bytes total memory
 53152  bytes free
```

CHKDSK will not correct the errors found in your directory unless you specify the /F (fix) switch. Typing /V causes CHKDSK to display messages while it is running.

you can redirect the output from CHKDSK to a file. Simply type:

```
CHKDSK A:> filename
```

The errors will be sent to the filename specified. Do not use the /F switch if you redirect CHKDSK output.

The following errors will be corrected automatically if you specify the /F switch:

Invalid drive specification

Invalid parameter

Invalid sub-directory entry

Cannot CHDIR to <filename>
Tree past this point not processed

First cluster number is invalid entry truncated

Allocation error, size adjusted

Has invalid cluster, file truncated

Disk error reading FAT

Disk error writing FAT

<filename> contains non-contiguous blocks

All specified file(s) are contiguous

You must correct the following errors returned by CHKDSK, even if you specified the /F switch:

Incorrect DOS version
: You cannot run CHKDSK on versions of MS-DOS that are not 2. or higher.

Insufficient memory
Processing cannot continue
: There is not enough memory in your machine to process CHKDSK for this disk. You must obtain more memory to run CHKDSK.

Errors found, F parameter not specified
Corrections will not be written to disk
: You must specify the /F switch if you want the errors corrected by CHKDSK.

Invalid current directory
Processing cannot continue
: Restart the system and re-run CHKDSK.

Cannot CHDIR to root
Processing cannot continue

The disk you are checking is bad. Try re-starting MS-DOS and RECOVER the disk.

<filename> is cross linked on cluster

Make a copy of the file you want to keep, and then delete both files that are cross linked.

X lost clusters found in y chains
Convert lost chains to files (Y/N)?

If you respond Y to this prompt, CHKDSK will create a directory entry and a file for you to resolve this problem (files created by CHKDSK are named FILEnnnnnnnn).

CHKDSK will then display:

X bytes disk space freed

If you respond N to this prompt and have not specified the /F switch, CHKDSK frees the clusters and displays:

X bytes disk space would be freed

Probable non-DOS disk
Continue (Y/N)?

The disk you are using is a non-DOS disk. You must indicate whether or not you want CHKDSK to continue processing.

Insufficient room in root directory
Erase files in root and repeat CHKDSK

CHKDSK cannot process until you delete files in the root directory.

Unrecoverable error in directory
Convert directory to file (Y/N)?

If you respond Y to this prompt, CHKDSK will convert the bad directory into a file. You can then fix the directory yourself or delete it.

CLS

Name: CLS

Type: Internal

Purpose: Clears the terminal screen.

Syntax: CLS

Comments: The CLS command causes MS-DOS to send the ANSI escape sequence ESC [2J (which clears your screen) to your console.

COMP

Name: COMP

Type: External

Purpose: Compares the contents of one set of files to another.

Syntax: [*drive*:] [*pathname*] COMP [*drive*:] [<*pathname1*>] [<*filespec1*>] [*drive*:] [<*pathname2*>] [<*filespec2*>]

Comments: COMP matches the first file against the second file and reports any differences between them. If only the drive number is specified as a second parameter, COMP assumes that the second filename is identical as the first one.

Pathnames can also be specified. For example:

```
COMP B:FOO \ BAR \ FILES1.TXT BAR \ FILE2.TXT
```

COMP takes FILE1.TXT in the \ FOO \ BAR directory of disk drive B: and compares it with FILE2.TXT in the \ BAR directory.
Since no drive is specified for filename2, COMP assumes that the \ BAR directory is on the disk in the default drive.

You can use the wild cards for both filenames, for example:

```
COMP B:*.TXT A:*.ASM
```

COMP compares the files which have file extension .TXT in the drive B with ones which have same file extension .ASM in drive A.

During comparison, COMP indicates the pathnames and filenames being compared. An error will occur in the following cases:

1. COMP cannot find a second set of files.
2. Two sets of files have different sizes.
3. A pathname you have entered is invalid.

COMP compares the two files byte-by-byte and if a mismatch is found, the following message will be displayed:

```
Compare error at offset XXXXXXXX
TEST1 = YY
TEST2 = ZZ
```

where filenames TEST1 and TEST2 are the filenames you had entered. XXXXXXXX is the relative address of the pair of bytes from the beginning of the file. YY and ZZ are the pair of mismatched bytes from TEST1 and TEST2 respectively.

When ten mismatchings occur within two files, COMP then terminates the comparison with the message:

```
10 Mismatches - ending compare
```

When COMP found that the contents of two files are identical,it displays:

```
Files compare ok

Compare more files (Y/N)?
```

Enter Y or N in response to this prompt.

In comparison, COMP always checks the end-of-file marks in the two files. If this mark is not found the error message:

```
Eof mark not found
```

will appear.

COPY

Name: COPY

Type: Internal

Purpose: Copies a file(s) to another disk with the same or different name. Appends or copies files on the same disk.

Syntax: COPY [<*drive:*>] <*pathname*> [<*drive:*>] [*pathname*] [/V] [/A] [/B]

Syntax: COPY <*pathname*> + <*pathname*>... <*pathname*>

Comments: To copy:

Note:

Complete pathnames are not necessary if the source and destination files are in the same working directory. You may use the filename only.

The copy will be placed on the default drive with the same name as the original file if the second pathname is not specified. An error will result if the original file is on the default disk and the second pathname is not specified, causing COPY to quit. MS-DOS will not allow files to be copied to themselves and will display the error message:

```
File cannot be copied onto itself
0 file(s) copied
```

There are three types of copy using the second option.

1. To specify the second option as a drive name only, will copy the file to the designated drive with the same file name as the original. For example, type:

```
COPY MEMO.DOC B:
```

will copy the file to the b: drive and name the file MEMO.DOC.

2. To specify the second option as a file name only, will copy the file to the default drive with the chosen name. For example, type:

COPY MEMO.DOC LETTER.DOC

will copy the file to the default drive and name the file LETTER.DOC.

3. To specify the second option as a drive name with a file name, will copy the file to the specified drive with the chosen name. For example, type:

COPY MEMO.DOC B:MEMO.DOC

will copy the file to B: drive and name the file MEMO.DOC.

Using switches:
To verify that the sectors were written correctly on the destination disk use the /V switch. MS-DOS will verify that critical data has been copied correctly, though there are few errors when copy is used. This switch will slightly slow down the copy process because a check must be performed on each disk entry. If there is a verification error a message will be displayed.

To process files as either ASCII (text) or binary use the /A and /B switches. The switch has control over the filename directly after the switch and all following filenames until another /A or /B switch is encountered.

When /A or /B switch is used with a source file name:

/A The file will be processed as an ASCII (text) file. The copying process will continue till the first control-Z (end-of-file mark). The control-Z nor the remainder of the file will not be copied.

/B The file will be copied in its entirety, including all control characters.

Examples:

```
COPY MEMO.TXT/A LETTER.TXT
COPY REPORT.ASM/B REPORT2.ASM
```

When used with the second filename option:

/A A control-Z (end-of-file) character is appended to the file.

/B A control-Z (end-of-file) character NOT appended to the file.

During the append process the default switch is always /A.

Examples:

```
COPY MEMO.TXT LETTER.TXT/A
COPY REPORT.ASM REPORT2.ASM/B
```

To Append:
To append files, use the copy command followed by a list of filenames separated by ' + ', then a destination file. The created file will be the accumulation of all specified files.

Example:

```
COPY INTRO.RPT + BODY.RPT + B:
SUM.RPT REPORT
```

The COPY command will combine the files of INTRO.RPT, BODY.RPT and on b: drive, SUM.RPT and place them in the file REPORT on the default drive.

By using a wild card you can combine several files into one file. Try typing

```
COPY *.LST COMBIN.PRN
```

and you will see that the COPY command selects all of the files with the extension .LST and combines them into the file COMBIN.PRN.
To demonstrate the power of this command, we will combine several matching files. Let's say we want all of the .LST files combined with all of the .REF files and place them in a .PRN file. Thus, FILE1.LST combines with FILE1.REF and are placed in FILE1.PRN. XYZ.LST combines with XYZ.REF and placed in XYZ.PRN, etc.

```
COPY *.LST + *.REF *.PRN
```

The Copy command can also combine all of the .LST and .REF files and place them in one file with the following, type:

```
COPY *.LST + *.REF COMBIN.PRN
```

Note:

An error will result if appending from a source file to a destination file that has the same name. For example:

```
COPY *.LST ALL.LST
```

If ALL.LST existed as a file before the copy command, the error will not be detected until ALL.LST is appended. By that time ALL.LST may have been destroyed.

During append the destination filename is compared with the input filename. If there is a match then the message "Content of destination lost before copy" is displayed and the input file is skipped and appending will continue normally. This permits the joining of files with the following command.

```
COPY ALL.LST+*.LST
```

The COPY command appends all files with the extension of .LST other than ALL.LST to ALL.LST.
No error message is displayed.

Messages:

Cannot do binary reads from a device
When copying from a device the binary mode can not be selected. Select the ASCII mode by specifying /A switch or remove /B switch.

Content of destination lost before copy
Before the copy procedure finished the source file was overwritten.

Example:

```
COPY X+Y Y
```

Y is destroyed before it can be copied.

CTTY

Name: CTTY

Type: Internal

Purpose: Allows you to change the device from which you issue commands (TTY represents the console).

Syntax: CTTY <*device*>

Comments: The <*device*> is the device from which you are giving commands to MS-DOS. This command is useful if you want to change the device on which you are working. The command

```
CTTY AUX
```

moves all command I/O (input/output) from the current device (the console) to the AUX port, such as a printer. The command

```
CTTY CON
```

moves I/O back to the original device (here, the console). Refer to the "Illegal Filenames" section of Chapter 3, "More About Files", for a list of valid device names to use with the CTTY command.

DATE

Name: DATE

Type: Internal

Purpose: Enter or change the date known to the system. This date will be recorded in the directory for any files you create or alter.

You can change the date from your terminal or from a batch file. (MS-DOS does not display a prompt for the date if you use an AUTOEXEC.BAT file, so you may want to include a DATE command in that file.)

Syntax: DATE [<*mm*>-<*dd*>-<*yy*>]

Comments: If you type DATE, DATE will respond with the message:

```
Current date is <mm>-<dd>-<yy>
Enter new date (mm-dd-yy):
```

Press <ENTER> if you do not want to change the date shown.

You can also type a particular date after the DATE command, as in:

```
DATE 3-9-85
```

In this case, you do not have to answer the Enter new date: prompt.

The new date must be entered using numerals only; letters are not permitted. The allowed options are:

<mm> = 1-12
<dd> = 1-31
<yy> = 80-99 or 1980-2099

The date, month, and year entries may be separated by hyphens (-) or slashes (/). MS-DOS is programmed to change months and years correctly, whether the month has 31,30,29,or 28 days. MS-DOS handles leap years,too.

If the options or separators are not valid, DATE displays the message:

```
Invalid date
Enter new date (mm-dd-yy):
```

DATE then waits for you to enter a valid date.

DEL

Name: DEL (DELETE)

Type: Internal

Synonym: ERASE

Purpose: Deletes all files with the designated filespec.

Syntax: DEL [*pathname*] [*filespec*]

Comments: If the filespec is *.*, the prompt Are you sure? appears. If a Y or y is typed as a response, then all files are deleted as requested. You can also type ERASE for the DELETE command.

DIR

Name: DIR (DIRECTORY)

Type: Internal

Syntax: DIR [*pathname*] [*filespec*] [/P] [/W]

Purpose: Lists the files in a directory.

Comments: If you just type DIR, all directory entries on the default drive are listed. If only the drive specification is given (DIR *d:*), all entries on the disk in the specified drive are listed. If only a filename is entered with no extension (DIR *filename*), then *all files* with the designated filename on the disk in the default drive are listed. If you designate a file specification (for example, DIR *d:filename.ext*), all *files* with the filename specified on the disk in the drive specified are listed. In all cases, files are listed with their size in bytes and with the time and date of their last modification.

The wild card characters ? and * (question mark and asterisk) may be used in the filename option. Note that for your convenience, the following DIR commands are equivalent:

COMMAND	EQUIVALENT
DIR	DIR *.*
DIR FILEName	DIR FILEName.*
DIR .EXT	DIR *.EXT

Two switches may be specified with DIR. The /P switch selects Page Mode. With /P, display of the directory pauses after the screen is filled. To resume display of output, press any key.

The /W switch selects Wide Display. With /W, only filenames are displayed, without other file information. Files are displayed five per line.

DISKCOMP

Name: DISKCOMP

Type: External

Purpose: Compares whole contents in one disk to another.

Syntax: [*drive*:] [*pathname*] DISKCOMP [*drive*:] [*drive*:] [/1] [/8]

Comments: Usually this command is used to confirm an identity between two disks after you copy a disk with DISKCOPY command.

The first option you specify is the source drive. The second option is the destination drive.

You can specify the same drives or you may specify different drives. If the drives designated are the same, a single-drive comparison is performed.

You are prompted to insert the disks at the appropriate times. DISKCOMP waits for you to press any key before continuing.

After no difference between two disks is found, DISKCOMP prompts:

Diskettes compare ok

Compare more diskettes (Y/N)?

If you press Y, the next comparison is performed on the same drives that you originally specified, after you have been prompted to insert the proper disks.

If any difference is found in one track, a message:

> **Compare error (s) on**
> **Track XX, side X**

will appear.

Notes:

1. If /1 option is specified, DISKCOMP compares single side on the disk.

2. The /8 option causes DISKCOMP to compare only 8 sectors per track, irrespective of the fact that the first disk has been formatted in 9 sectors per track.

DISKCOPY

Name: DISKCOPY

Type: External

Purpose: Copies the contents of the disk in the source drive to the disk in the destination drive.

Syntax: [*drive*:] [*pathname*] DISKCOPY [*drive*:] [*drive*:] [*/1*]

Comments: The first option you specify is the source drive. The second option is the destination drive.

You can specify the same drives or you may specify different drives. If the drives designated are the same, a single-drive copy operation is performed. You are prompted to insert the disks at the appropriate times. DISKCOPY waits for you to press any key before continuing.

After copying, DISKCOPY prompts:

```
Copy complete
Copy another (Y/N)?
```

If you press Y, the next copy is performed on the same drives that you originally specified, after you have been prompted to insert the proper disks.

To end the copy, press N.

Notes:

1. If you omit both options, a single drive copy operation will be performed on the default drive.

2. If you omit the second option, the default drive will be used as the destination drive.

3. If the /1 option is specified, DISKCOPY copies only one side of the disk.

4. Both disks must have the same number of physical sectors and those sectors must be the same size.

5. If the disk in the destination drive has not been formatted with the same number of sides and sectors per track as the one in the source drive, DISKCOPY will format the disk in the destination drive during copy operation prompting that:

   Formatting while copying

6. Disks that have a lot of file creation and deletion activity become fragmented, because disk space is not allocated sequentially. The first free sector found is the next sector allocated, regardless of its location on the disk.

   A fragmented disk can cause poor performance due to delays involved in finding, reading, or writing a file. If this is the case, you must use the copy command, instead of DISKCOPY, to copy your disk and eliminate the fragmentation.

   For example:

   COPY A:*.* B:

Copies all files from the disk in drive A: to the disk in drive B:.

7. DISKCOPY automatically determines the number of sides to copy, based on the source drive and disk.

8. If disk errors are encountered during a DISKCOPY, MS-DOS displays:

DISK error while reading drive A

Abort, Retry, Ignore?

Refer to Appendix B, Disk Errors, for information on this error message.

EXE2BIN

Name: EXE2BIN

Type: External

Purpose: Converts .EXE (executable) files to binary format. This results in a saving of disk space and faster program loading.

Syntax: [*drive*:] [*pathname*] EXE2BIN <*filespec*> [*drive*:] [<*filename*> [<*.ext*>]]

Comments: This command is useful only if you want to convert .EXE files to binary format. The file named by filespec is the input file. If no extension is specified, it defaults to .EXE. The input file is converted to .COM file format (memory image of the program) and placed in the output file. If you do not specify a drive, the drive of the input file will be used. If you do not specify an output filename, the input filename will be used. If you do not specify a filename extension in the output filename, the new file will be given an extension of .BIN.

The input file must be in valid .EXE format produced by the linker. The resident, or actual code and data part of the file must be less than 64K. There must be no STACK segment.

Two kinds of conversions are possible, depending on whether the initial CS:IP (Code Segment: Instruction Pointer) is specified in the .EXE file:

1. If CS:IP is not specified in the .EXE file, a pure binary conversion is assumed. If segment fixups are necessary (i.e., the program contains instructions requiring segment relocation), you will be prompted for the fixup value. This value is the absolute segment at which the program is to be loaded. The resulting program will be usable only when loaded at the absolute memory address specified by a user application. The command processor will not be capable of properly loading the program.

2. If CS:IP is specified as 0000:100H, it is assumed that the file is to be run as a .COM file with the location pointer set at 100H by the assembler statement ORG; the first 100H bytes of the file are deleted. No segment fixups are allowed, as .COM files must be segment relocatable; that is, they must assume the entry conditions explained in the *Macro Assembler Manual*. Once the conversion is complete, you may rename the resulting file with a .COM extension. Then the command processor will be able to load and execute the program in the same way as the .COM programs supplied on your MS-DOS disk.

If CS:IP does not meet either of these criteria, or if it meets the .COM file criterion but has segment fixups, the following message will be displayed:

File cannot be converted

This message is also displayed if the file is not a valid executable file.

If EXE2BIN finds an error, one or more of the following error messages will be displayed:

File not found
The file is not on the disk specified.

Insufficient memory
There is not enough memory to run EXE2BIN.

File creation error
EXE2BIN cannot create the output file. Run CHKDSK to determine if the directory is full, or if some other condition caused the error.

Insufficient disk space
There is not enough disk space to create a new file.

Fixups needed - base segment (hex):
The source (.EXE) file contained information indicating that a load segment is required for the file. Specify the absolute segment address at which the finished module is to be located.

File cannot be converted
The input file is not in the correct format.

WARNING -Read error on EXE file.
Amount read less than size in header
This is a warning message only.

EXIT

Name: EXIT

Type: Internal

Purpose: Exits the program COMMAND.COM (the command processor) and returns to a previous level, if one exists.

Syntax: EXIT

Comments: This command can be used when you are running an application program and want to start the MS-DOS command processor, then return to your program. For example, to look at a directory on drive B: while running an application program, you must start the command processor by typing COMMAND in response to the default drive prompt:

```
A>COMMAND
```

You can now type the DIR command and MS-DOS will display the directory for the default disk. When you type EXIT, you return to the previous level (your application program).

FIND

Name: FIND

Type: External

Purpose: Searches for a specific string of text in a file or files.

Syntax: [*drive*:] [*pathname*] FIND [/V/C/N] <*string*> [<*filename*... >]

Comments:

FIND is a filter that takes a string and, as options, a series of filenames. It will display all lines that contain a specified string from the files specified in the command line.

If no files are specified, FIND will take the input on the screen and display all lines that contain the specified string.

Switches for FIND are:

- /V causes FIND to display all lines not containing the specified string.
- /C causes FIND to print only the count of lines that contained a match in each of the files.
- /N causes each line to be preceded by its relative line number in the file.

The string should be enclosed in quotes.
Example:

```
FIND "Fool's Paradise"BOOK1.TXT BOOK2.TXT
```

displays all lines from BOOK1.TXT and BOOK2.TXT (in that order) that contain the string "Fool's Paradise". The command

```
DIR B: |FIND / V"DAT"
```

causes MS-DOS to display all names of the files on the disk in drive B: which do not contain the string DAT. Type double quotes around a string that already has quotes in it.

When an error is detected, FIND responds with one of the following error messages:

Incorrect DOS version
FIND will only run on versions of MS-DOS that are 2.0 or higher.

FIND: Invalid number of parameters
You did not specify a string when issuing the FIND command.

FIND: Syntax error
You typed an illegal string when issuing the FIND command.

FIND: File not found <filename>
The filename you have specified does not exist or FIND cannot find it.

FIND: Read error in <filename>
An error occurred when FIND tried to read the file specified in the command.

FIND: Invalid Parameter <option-name>
You specified an option that does not exist.

FORMAT

Name: FORMAT

Type: External

Purpose: Initializes the floppy disk with a recording format that is acceptable to DOS in the default or designated drive. Initializes the directory area for DOS files, File Allocation Table and system loader and checks the disk for defective tracks.

Warning:
Care should be taken when using the FORMAT command, because FORMAT destroys all previous information on the disk. This is true for hard disks too, for a DOS partition, which includes subdirectories etc. will be erased. When using FORMAT it is wise to specify the drive that is to be formatted. Otherwise, the default drive is assumed to be specified and if you are not sure which drive is the default drive you could unintentionally destroy valuable information.

Syntax: [*drive:*] [*pathname*] FORMAT [*drive:*] [/S] [/1] [/8] [/V] [/B] [/4]

Comments: [*drive:*] [*pathname*] before FORMAT determines the drive and the path of the FORMAT command file.

[*drive:*] specifies the drive, floppy disk or hard disk that is to be formatted.

/S copies the operating system after formatting the disk. The operating system needs to be on the default drive. If the default drive is a hard disk drive then DOS will prompt you to insert the operating system into A drive or if the default drive is a floppy disk drive and the system disk is absent from the disk, then DOS will prompt you to insert a disk containing the operating system. The operating system files are copied in the following order:

IO.SYS
MSDOS.SYS
COMMAND.COM

/1 causes the floppy disk to be formatted for single sided use regardless of the drive type.

/8 causes a floppy disk to be formatted for 8 sectors per track. FORMAT defaults to 9/15 sectors per track use if you do not specify /8. FORMAT always creates 9/15 physical sectors on each disk track, but it instructs DOS to use only 8 sectors per track if you use the /8 switch.

/V labels each disk with a volume name. It has no other function, but is very useful for cataloging your floppy disks.

/B sets the number of sectors per track to 8 with space allocated for the IO.SYS and MSDOS.SYS system files. The command file nor the system files are placed on the disk. The space provided can be used by any version of DOS (1.00, 1.10, 2.10, or 3.00) with the SYS command. When the /B switch is omitted only version 3.00 may be placed on the floppy disk with the SYS command.

with /4 switch, in a high capacity drive, a double sided floppy disk can be formatted.

Caution:

Care should be taken when using the /4 switch. The ability to read and write with a single or double sided drive may not be reliable.

Valid switches for different disk types are shown in the table below.

Disk Type	Parameters Allowed
320KB	/S, /V, /1, /8, /B, /4
1.2MB	/S, /V
Hard Disk	/S, /V

Notes:

1. To use disk media, it must be formatted before use by DOS.

2. To change the size of the DOS partition on a hard disk using the FDISK command, the hard disk must be formatted again.

3. Existing information on a disk is erased by formatting.

4. Defective tracks found during formatting are marked as 'reserved'. The tracks will not be used to store information.

5. The two system files, IO.SYS and MSDOS.SYS are hidden and will not be listed on a directory search, such as DIR command.

6. When using the /V switch of the FORMAT command, a prompt will be displayed requesting the volume label. The label can be 11 characters long or less and is comprised of all characters acceptable as filenames except that a period is not necessary as an extension delimiter.
   Volume labels may be changed or added by using the LABEL command. For more information please refer to the description of the LABEL command in this manual.

7. A status report is generated by FORMAT, that displays:

   - Total disk space
   - Defective space that has been marked
   - The DOS system file allocated space (/S used)
   - User available space

8. FORMAT will automatically determine the specified drive type and format the disk or floppy disk accordingly. A floppy disk formatted as single sided can be read or written on only one side and can be used in either drive type. A floppy disk formatted without the /1 switch can be read or written on both sides, but cannot be used in a single-sided drive.

9. Hard disks are normally preformatted before being shipped. FORMAT only checks the DOS partition locations, a physical format is not performed.

10. When using the /S switch the system files are loaded into memory. If available memory is not sufficient several disk reads and writes operations will be performed after the disk formatting process to transfer the system files onto the disk being formatted. A message to reinsert the disk will be displayed if the disk is removed before the completion of the formatting process.

11. When using the /B switch, the /8 or the /V switch cannot be used.

12. When using the /S switch, and the default drive is a hard disk that does not have the system files, a message will be displayed to insert a systems disk in A drive.

Types of Floppy Disk Drives

The following floppy disk drives can be used with your computer.

- Single Sided (160KB/180KB)
- Double Sided (320KB/360KB)
- High Capacity (1.2MB)

Types of Floppy Disks

The reading and writing of information can be performed on the following floppy disks.

- Single Sided (160KB/180KB)
- Double Sided (320KB/360KB)
- High Capacity (1.2MB)

A single sided floppy disk has 40 tracks with 8 or 9 sectors per track, a maximum capacity of 160K or 180K (one K is 1024) bytes.

A double sided floppy disk has 40 tracks with 8 or 9 sectors per track, a maximum capacity of 320K or 360K (one K is 1024) bytes.

A high capacity floppy disk is double sided having 80 tracks with 15 sectors per track, a capacity of 1.2M (one M is 1,048,576) bytes.

Floppy Disks and Drive Compatibility

Some floppy disk types and drive types are not compatible for read and write operations. The section below describes the compatibility of each combination.

Single Sided Drives:

Read and write may be performed to:

- Single Sided Floppy Disks

Double Sided Drives:
Read and write may be performed to:

- Single Sided Floppy Disks
- Double Sided Floppy Disks

High Capacity Drives:
Read and write may be performed to:

- Single Sided Floppy Disks
- Double Sided Floppy Disks
- High Capacity Floppy Disks

Caution:

Single or double sided drives may not be able to read a floppy disk that has been written on in a high capacity drive.

Consideration of floppy disk and drive compatibility are needed when system commands that read and write to the floppy disk are used. For example, FORMAT is an external command and the section "FORMAT Compatibility" should be read before using the command.

Format Compatibility

Acceptable FORMAT combinations are shown in the table below:

Drive Type	Floppy Disk Type
Single Sided	Single sided floppy disks
Double Sided	Single sided or double sided floppy disks
High Capacity	Single sided*, double sided* or high capacity floppy disks

* Refer to the FORMAT command using /4 switch when formatting single or double sided floppy disks in a high capacity drive.

Example: In the command shown below, the floppy disk in B drive is formatted, the system files are transferred and a request for the volume label is displayed.

```
A>FORMAT B:/S/V
```

The following message is displayed by the system.

```
Insert new diskette for drive B:
and strike ENTER when ready
```

Insert the requested floppy disk and press <ENTER>, then the following message is displayed

```
Formatting...
```

during the formatting process.

The next message is displayed after the formatting process.

```
Formatting...Format complete
System transferred

Volume label (11 characters, ENTER for none)? MYDISK

    xxxxxx bytes total disk space
     xxxxx bytes used by system
    xxxxxx bytes available on disk

Format another (Y/N)? n
```

Please note that the volume label chosen was MYDISK.
To format another disk, type Y and press <ENTER>. To end the formatting program, type N and press <ENTER>.

The following prompt is displayed when a hard disk is formatted.

Warning, All Data on Non-Removable
Disk Drive x: Will Be Lost!
Proceed with Format (Y/N)?

The drive letter you type will replace the x. If Y is typed and <ENTER> is pressed, then your hard disk will be formatted. If N is typed and <ENTER> is pressed then the format program will end.

Formatting a hard disk takes several minutes because of its large capacity.
To check if formatting is still processing see if the indicator lamp is illuminated on the hard disk drive. After formatting is completed the prompt for the volume label is displayed.

GRAPHICS

Name: GRAPHICS

Type: External

Purpose: Prints the contents which is currently displayed on a screen to a printer.

Syntax: [*drive*:] [*pathname*] GRAPHICS

Comments: This command is applicable for the color version computer. When the screen is in the graphic mode, pressing the PrtSc (print screen) key while the SHIFT key is held down allows you to print out the contents on the screen.

Notes:

1. When the screen is in the 320 x 200 pixels color graphic mode, difference in color is expressed in four shaded patterns on the paper.

2. When the screen is in the 640 x 200 pixels color graphic mode, the contents of the screen is printed sideway on the paper. The upper right corner of the screen is printed on the upper left corner of the paper.

JOIN

Name: JOIN

Type: External

Purpose: Joins a disk drive to a specific pathname.

Syntax: [*drive:*] [*pathname*] JOIN <*drive:*> <*drive:*> <*pathname*>

Comments: MS-DOS will attempt to create a directory with the pathname if the specified pathname does not exist. The path must be empty. The JOIN command will make the first drive name invalid and display the error message, "Invalid drive" if it is used.

The distinction of the drive letter that is used to address the physical drives is removed and all directories can be referred to by one pathname on a specific drive. A previously existing path will not be usable while the JOIN command is in effect.

The root path of a drive can only be joined. The command is valid:

```
JOIN D: C:\MEMOS
```

The following command is not valid:

```
JOIN D: C:\MEMOS JUNE
```

To release the join, use the following format:

```
JOIN <drive:> /D
```

<*drive:*> is the source drive and the /D switch releases the JOIN command.

To display the currently joined drives type:

```
JOIN
```

Messages:

Directory not empty
The directory to be joined to must be empty.

Incorrect number of parameters
Too few or too many options were specified in the command line.

Not enough memory
Not enough memory is available for MS-DOS to run the command.

LABEL

Name: LABEL

Type: External

Purpose: Used for creating, changing, or deleting a volume label on a disk.

Syntax: [*drive:*] [*pathname*] LABEL [*drive:*] [*volume label*]

Comments: Specify the following parameters:

[*drive:*] [*pathname*] is entered before LABEL to specify the drive and path containing the LABEL command file.

[*drive:*] specifies the drive letter of the disk to be labeled.

[*volume label*] specifies the volume label. Each disk is identified by a volume label. Up to 11 characters can be used in specifying the label and the format is the same as that for volume labels created by FORMAT/V. The following prompt appears if you do not specify a volume label:

Volume label (11 characters, ENTER for none)?

Select and type the desired volume label for the disk and press <ENTER>.

To change a current volume label, type the new label desired and press <ENTER>.

To delete a current volume label, do not specify a volume label. By just pressing <ENTER>, the current volume label on the disk will be deleted.

If more than 11 characters are entered for the volume label, only the first 11 characters will be used.

Example: This example shows how to create a volume label called ADDRESS on the disk in drive A.

```
A>LABEL A:ADDRESS
```

This example illustrates creating the volume label HARDDISKC for the hard disk C.

```
A>LABEL C:HARDDISKC
```

This example shows how to change the volume label of the disk in drive A to PROGRAMS.

```
A>LABEL A:
```

Then the following prompt appears:

```
Volume label (11 characters, ENTER for none)?
```

Type the following:

```
PROGRAMS
```

To delete the volume label PROGRAMS from the disk in drive A, follow this example.

```
A>LABEL A:
```

Then the following prompt will appear:

```
Volume label (11 characters, ENTER for none)?
```

The volume label PROGRAMS will be deleted when you press <ENTER>.

MKDIR

Name: MKDIR

Type: Internal

Synonym: MD

Purpose: Makes a new directory.

Syntax: MKDIR <*pathname*>

Comments: This command is used to create a hierarchical directory structure. When you are in your root directory, you can create subdirectories by using the MKDIR command. The command

```
MKDIR \USER
```

will create a subdirectory \USER in your root directory. To create a directory named JOE under \USER, type:

```
MKDIR \USER \JOE
```

MODE

Name: MODE

Type: External

Purpose: Sets the mode of operation on a printer, display monitor, RS-232C port and so on.

Syntax: [*drive:*] [*pathname*] MODE LPT#: [*n*:] [, [*m*] [,P]] (for printer)

MODE *n* or MODE *n,m* [,T] (for display monitor)

MODE COM*n*: *baudrate* [,*parity* [,*databits* [,*stopbits* [,P]]]]
(for RS-232C port)

MODE LPT#:= COM*n* (for redirection from printer output to RS-232C port)

Comments: **Format 1 (for printer)**

MODE LPT #: [*n*] [,[*m*] [,P]]

where:
: printer number, should be 1, 2 or 3
n : number of printable characters per line, should be 80 or 132.
p : Specifies continuous retry on time-out errors.

Format 2 (for display monitor)

MODE [*n*], *m* [,T]

where:
n : should be 40, 80, BW40, BW80, CO40, CO80, or MONO

Description of each parameter is as follows. The parameters other than MONO are applicable for the computer which has color display board.

40: sets the screen width to 40 characters per line.

80: sets the screen width to 80 characters per line.

BW40: activates the color display board and sets it to black and white display mode with 40 characters per line.

BW80: activates the color display board and sets it to black and white display mode with 80 characters per line.

CO40: activates the color display board and sets it to color disp.ay mode with 40 characters per line.

CO80: activates the color display board and sets it to color display mode with 80 characters per line.

MONO: activates the monochrome display board. Screen width cannot be switched.

m : should be R or L. It shifts the screen to the right or left.

T : displays test characters.

Format 3 (for RS-232C port)

MODE COM*n*: *baudrate* [,*parity* [,*databits* [,*stopbits* [,P]]]]

Where:

n : should be 1 or 2. This determines port number.

baudrate: 110, 150, 300, 600, 1200, 2400, 4800 or 9600. When specifying,input first two characters of these numbers.

parity: should be N (none), O (odd), or E (even). The default value is E.

databits: should be 7 or 8. The default value is 7.

stopbits: should be 1 or 2. When the baudrate is 110, it defaults 2 and other baudrate, it defaults 1.

P : specifies when using the RS-232C port as a printer interface.

Format 4 (for redirecting printer output to RS-232C port)

MODE LPT#:=COM*n*

where:

: should be 1, 2 or 3. This determines printer number.

n : should be 1 or 2. This determines RS-232C port number.

All output directed to printer will be redirected the RS-232C port. Before you use this format of the MODE command, you should initialize communication protocol by referring to the Format 3 above with assigning P option.

MORE

Name: MORE

Type: External

Purpose: Sends output to console one screen at a time.

Syntax: [*drive*:] [*pathname*] MORE

Comments: MORE is a filter that reads from standard input (such as a command from your terminal) and displays one screen of information at a time. The MORE command then pauses and displays the --MORE-- message at the bottom of your screen.

Pressing the <ENTER> key will display another screen of information. This process continues until all the input data has been read.

The MORE command is useful for viewing a long file one screen at a time. If you type:

```
TYPE MYFILES.COM|MORE
```

MS-DOS will display the file MYFILES.COM (on the default drive) one screen at a time.

PARK

Name: PARK

Type: External

Purpose: Moves the read and write head on a hard disk to safety zone to avoid possible damage.

Syntax: [*drive*:] [*pathname*] PARK

Comments: Before you move your hard disk system use this command to prevent possible damage to the head.

To prevent data loss from a minor shock when moving your system, the PARK command places the read/write head in an area that is least likely to cause damage.

At the A> prompt type

PARK

and press <ENTER>.

Your screen will display the following message.

HARD DISK HEAD PARK PROGRAM

Remove disks and turn off the power
or press [ESC] to return to system...

You can now turn the power off on your hard disk drive. If you wish to continue press the [ESC] key.

Note:

The error message

No hard disk present

will be displayed on your screen if your system is not equipped with a hard disk drive.

If the hard disk controller malfunctions the following message is displayed.

Hard disk controller error

PATH

Name: PATH

Type: Internal

Purpose: Sets a command path.

Syntax: PATH [<*pathname*>[;<*pathname*>]...]

Comments: This command allows you to tell MS-DOS which directories should be searched for external commands after MS-DOS searches your working directory. The default value is no path.

To tell MS-DOS to search your \ BIN \ USER \ JOE directory for external commands, type:

```
PATH \BIN\USER\JOE
```

MS-DOS will now search the \ BIN \ USER \ JOE directory for external commands until you set another path or shut down MS-DOS.

You can tell MS-DOS to search more than one path by specifying several pathnames separated by semicolons. For example,

```
PATH \BIN\USER\JOE;\BIN\USER\SUE;\BIN\DEV
```

tells MS-DOS to search the directories specified by the above pathnames to find external commands. MS-DOS searches the pathnames in the order specified in the PATH command.

The command PATH with no options will print the current path. If you specify PATH;, MS-DOS will set the NUL path, meaning that only the working directory will be searched for external commands.

PRINT

Name: PRINT

Type: External

Purpose: Permits the processing of other MS-DOS commands while sending a text file to a line printer for printing. This is referred to as "background printing"

Syntax: [*drive*:] [*pathname*] PRINT [*drive*:] [*pathname*] [/D:<*device*>] [/B:<*size*>] [/U:<*value*>] [/M:<*value*>] [/S:<*value*>] [/Q:<*value*>] [/T] [/C] [/P]

Comments: A line printer that is attached to your computer requires the use of this command. The switches described below are used with this command:

/D To specify the print device. The default device is PRN if the switch is not used, and a prompt will be displayed.

The following switches are used to configure the PRINT command and are executed only the first time the PRINT command is used after MS-DOS is started.

/B The byte size of the internal buffer is set with this switch. The speed of the print command increases with a higher value.

/U The clock cycles of your computer that the print command will wait for the printer to become available. The time slice is given up if the PRINT command waits longer than the specified value. Refer to /S switch below.

/M The clock cycles of your computer for the PRINT command to print can be specified. A value from 1 to 255 may be specified or the default clock cycles is 2.

/S The time slice value is specified, or the default is 8.

/Q The number of files permissible in the print queue is specified. The maximum is 32 files, the minimum is 4 files, and the default is 10 files.

/T The deletion of all files waiting in the print queue are deleted and a message acknowledging this is displayed.

/C Cancel mode is turned on and removes the preceding filenames and all following filenames from the print queue.

/P Print mode is turned on and adds the preceding filenames and all following filenames to the print queue.

To display the filenames waiting in the print queue, type

PRINT

or type

PRINT /T

to remove all files from the print queue.

PRINT A:TEMP1.TST/C A:TEMP2.TST A:TEMP3.TST

The three files in the command line above are removed from the print queue.

PRINT TEMP1.TST/C TEMP2.TST/P TEMP3.TST

TEMP1.TST is removed from the print queue, and TEMP2.TST and TEMP3.TST are added to the print queue.

Messages:

All files canceled
MS-DOS will print the message "All files canceled by operator" on your printer if the /T switch is used. The message "File canceled by operator" will be printed if the /C switch is used.

Cannot open (filename)
The specified file can not be found or does not exist. The command line should be checked for a valid filename.

Errors on list device indicate that it may be offline. Please check it
The printer is not turned on.

(filename) file not found
The file that was queued did not start printing when the floppy disk was changed. Type the print command for that file again.

List output is not assigned to a device
The specified list device is invalid. A valid list device needs to be specified in the print command line.

Name of list device [PRN:]
When the print command is run the first time and the /D switch was not specified this message is displayed. The list device for print output can be any current device or by pressing <ENTER> MS-DOS will expect the PRN: device as default.

No files match drive: xxxxxxxx.xxx
The specified drive and filename were not found and cannot be added to the print queue.

Note:

An error message is not displayed if files in the print queue do not match the canceled file.

Print queue is empty
The print queue does not contain any files.

Print queue is full
You have added more than the maximum specified files in the print queue. The default is 10 files or refer to /Q switch in the list above.

Resident part of Print installed
MS-DOS displays this message the first time the print command is used. To process this command and others the user available memory is reduced by several thousand bytes.

PROMPT

Name: PROMPT

Type: Internal

Purpose: Changes the MS-DOS command prompt.

Syntax: PROMPT [<*prompt-text*>]

Comments: This command allows you to change the MS-DOS system prompt (for example, A>). If no text is typed, the prompt will be set to the default prompt, which is the default drive designation. You can set the prompt to a special prompt, such as the current time, by using the characters indicated below.

The following characters can be used in the prompt command to specify special prompts. They must all be preceded by a dollar sign ($) in the prompt command:

Specify This Character		To Get This Prompt:
$	—	The '$' character
t	—	The current time
d	—	The current date
p	—	The current directory of the default drive
v	—	The version number
n	—	The default drive
g	—	The '>' character
l	—	The '<' character
b	—	The '|' character
_	—	A CR/LF sequence
s	—	A space (leading only)
h	—	A backspace
e	—	ASCII code X'1B' (escape)

Examples:

PROMPT $n
Sets the normal MS-DOS prompt (>).

PROMPT Time = t_Date = $d
Sets a two-line prompt which prints:
Time = (current time)
Date = (current date)

If your terminal has an ANSI escape sequence driver, then you can use escape sequences in your prompts. For example:

PROMPT $e[7m$n:$e[m
Sets the prompts in inverse video mode and returns to video mode for other text.

RECOVER

Name: RECOVER

Type: External

Purpose: Recovers a file or an entire disk containing bad sectors.

Syntax: [*drive*:] [*pathname*] RECOVER <*filename*|*d*:>

Comments: If a sector on a disk is bad, you can recover either the file containing that sector (without the bad sector) or the entire disk (if the bad sector was in the directory).

To recover a particular file, type:

RECOVER <filename>

This will cause MS-DOS to read the file sector by sector and to skip the bad sector(s). When MS-DOS finds the bad sector(s), the sector(s) are marked and MS-DOS will no longer allocate your data to that sector.

To recover disk, type:

RECOVER <*drive*:>

where d: is the letter of the drive containing the disk to be recovered.

If there is not enough room in the root directory, RECOVER will print a message and store information about the extra files in the File Allocation Table. You can run RECOVER again to regain these files when there is more room in the root directory.

REM

Name: REM (REMARK)

Type: Internal

Purpose: Displays remarks which are on the same line as the REM command in a batch file during execution of that batch file.

Syntax: REM [*comment*]

Comments: The only separators allowed in the comment are the space, tab, and comma.

Example:

```
1: REM This file checks new disks
2: REM It is named NEWDISK.BAT
3: PAUSE Insert new disk in drive B:
4: FORMAT B:/S
5: DIR B:
6: CHKDSK B:
```

REN

Name: REN (RENAME)

Type: Internal

Synonym: RENAME

Purpose: Changes the name of the first option (*filespec*) to the second option (*filename*).

Syntax: REN <*filespec*> <*filename*>

Comments: The first option (*filespec*) must be given a drive designation if the disk resides in a drive other than the default drive. Any drive designation for the second option (*filename*) is ignored. The file will remain on the disk where it currently resides.

The wild card characters may be used in either option. All files matching the first filespec are renamed. If wild card characters appear in the second filename, corresponding character positions will not be changed.

For example, the following command changes the names of all files with the .LST extension to similar names with the .PRN extension:

```
REN *.LST *.PRN
```

In the next example, REN renames the file ABODE on drive B: to ADOBE:

```
REN B:ABODE ?D?B?
```

The file remains on drive B:.

An attempt to rename a filespec to a name already present in the directory will result in the error message "Duplicate file name" or "File not found."

RESTORE

Name: RESTORE

Type: External

Purpose: Restores files that were backed up by the Microsoft or IBM backup program.

Syntax: [*drive*:] [*pathname*] RESTORE <*drive*:>[*drive*:] [*pathname*] [/S] [/P] [/B:<*date*>] [/A:<*date*>] [/E:<*time*>] [/L:<*time*>] [/M] [/N]

Comments: The drive name specified first after RESTORE is the drive that contains the disk with the backed up files. The drive and pathname of the desired files are specified in the next two options.

The restore program supplied by IBM and this program are compatible except for the inclusion of the following switches.
The MS-DOS restore command uses the following switches.

/S Subdirectories are also restored.

/P Give a prompt for read-only or hidden files that are matched.

/B The files on or before the specified date are restored if they were modified.

/A The files on or after the specified date are restored if they were modified.

/E The files at or before the specified time are restored if they were modified.

/L The files at or after the specified time are restored if they were modified.

/M The files since the last backup are restored if they were modified.

/N The files that do not exist on the destination disk are restored.

The following ERRORLEVEL is set by the backup program.

0 Normal completion
1 No files were found to back up
3 Terminated by user
4 Terminated due to error

RMDIR

Name: RMDIR (REMOVE DIRECTORY)

Type: Internal

Synonym: RD

Purpose: Removes a directory from a hierarchical directory structure.

Syntax: RMDIR <*pathname*>

Comments: This command removes a directory that is *empty* except for the . and .. shorthand symbols.

To remove the \ BIN \ USER \ JOE directory, first issue a DIR command for that path to ensure that the directory does not contain any important files that you do not want deleted. Then type:

```
RMDIR \BIN\USER\JOE
```

The directory has been deleted from the directory structure.

SET

Name: SET

Type: Internal

Purpose: Sets one string value equivalent to another string for use in later programs.

Syntax: SET [<*string* = *string*>]

Comments: This command is meaningful only if you want to set values that will be used by programs you have written. An application program can check all values that have been set with the SET command by issuing SET with no options. For example, SET TTY = VT52 sets your TTY value to VT52 until you change it with another SET command.

The SET command can also be used in batch processing. In this way, you can define your replaceable parameters with names instead of numbers. If your batch file contains the statement "LINK %FILE%", you can set the name that MS-DOS will use for that variable with the SET command. The command SET FILE = DOMORE replaces the %FILE% parameter with the filename DOMORE. Therefore, you do not need to edit each batch file to change the replaceable parameter names. Note that when you use *text* (instead of numbers) as replaceable parameters, the name must be ended by a percent sign.

SHARE

Name: SHARE

Type: External

Purpose: Installs locking and file sharing.

Syntax: [*drive:*] [*pathname*] SHARE [/F:<*space*>] [/L:<*locks*>]

Comments: This command is only used when networking is active. It is used to install shared files and is included in the AUTOEXEC.BAT file. To learn more about shared files see "Microsoft Networks Manager's Guide"

The area size (in bytes) used by MS-DOS to record filesharing information is allocated by the [/F:<*space*>] switch. Full filenames plus 11 bytes for each file that is opened is needed with the average pathname at 20 bytes. 2048 bytes is the default value.

The number of locks may be allocated by the [L:<*lock*>] switch. 20 locks is the default value.

A check is made on all read and write requests after the SHARE command has been used.

Example: File sharing is loaded with the default values of switches /F and /L.

```
SHARE
```

Messages:

Incorrect parameter
An error occurred in a specified option.

Not enough memory
MS-DOS can not run the command with the amount of existing memory.

Share Already Installed
You can not install SHARE more than once.

SORT

Name: SORT

Type: External

Purpose: SORT reads input from your terminal, sorts the data, then writes it to your terminal screen or files.

Syntax: [*drive*:] [*pathname*] SORT [/R] [/+*n*]

Comments: SORT can be used, for example, to alphabetize a file by a certain column. There are two switches which allow you to select options:

/R reverse the sort; that is, sort from Z to A.

/+n sort starting with column n where n is some number. If you do not specify this switch, SORT will begin sorting from column 1.

Examples:

This command will read the file UNSORT.TXT, reverse the sort, and then write the output to a file named SORT.TXT:

```
SORT /R <UNSORT.TXT> SORT.TXT
```

The following command will pipe the output of the directory command to the SORT filter. The SORT filter will sort the directory listing starting with column 14 (this is the column in the directory listing that contains the file size), then send the output to the console. Thus, the result of this command is a directory sorted by file size:

```
DIR|SORT/+14
```

The command

```
DIR | SORT / +14 | MORE
```

will do the same thing as the command in the previous example, except that the MORE filter will give you a chance to read the sorted directory one screen at a time.

SUBST

Name: SUBST

Type: External

Purpose: Substitutes a string alias for a pathname.

Syntax: [*drive*:] [*pathname*] SUBST [<*drive*:>] [*pathname*] [/D]

Comments: By associating a pathname with a drive letter a virtual drive can be created or deleted with the SUBST command.
A new pathname replaces the previous reference to a virtual drive by MS-DOS when it is created by the SUBST command.
By typing

SUBST

MS-DOS displays the current names of the virtual drives.
The /D switch is used to delete an associated drive or pathname.

Examples: Type the command

SUBST Z: B: \ USR \ FRED \ FORMS

we have created a virtual Z drive for the pathname "B: \ USR \ FRED \ FORMS". Now, to reach this directory you only need to type Z:

Messages:

Incorrect number of parameters
Too few or too many options were specified in the command line.

Not enough memory
MS-DOS can not run the command with the amount of existing memory.

SYS

Name: SYS (SYSTEM)

Type: External

Purpose: Transfers the MS-DOS system files from the disk in the default drive to the disk in the drive specified by d:.

Syntax: [*drive*:] [*pathname*] SYS <*drive*:>

Comments: SYS is normally used to update the system or to place the system on a formatted disk which contains no files. An entry for d: is required.

If IO.SYS and MSDOS.SYS are on the destination disk, they must take up the same amount of space on the disk as the new system will need. This means that you cannot transfer system files from an MS-DOS 3.0 disk to an MS-DOS1.1 disk. You must reformat the MS-DOS 1.1 disk with the MS-DOS FORMAT command before the SYS command will work.

The destination disk must be completely blank or already have the system files IO.SYS and MSDOS.SYS

The transferred files are copied in the following order:

IO.SYS
MSDOS.SYS

IO.SYS and MSDOS.SYS are both hidden files that do not appear when the DIR command is executed. COMMAND.COM (the command processor) is *not* transferred. You must use the COPY command to transfer COMMAND.COM.

If SYS detects an error, one of the following messages will be displayed:

No room for system on destination disk

There is not enough room on the destination disk for the IO.SYS and MSDOS.SYS files.

Incompatible system size

The system files IO.SYS and MSDOS.SYS do not take up the same amount of space on the destination disk as the new system will need.

TIME

Name: TIME

Type: Internal

Purpose: Displays and sets the time.

Syntax: TIME [<*hh*> [:<*mm*>]]

Comments: If the TIME command is entered without any arguments, the following message is displayed:

```
Current time is <hh>:<mm>:<ss>.<cc>
Enter new time:_
```

Press the <ENTER> key if you do not want to change the time shown. A new time may be given as an option to the TIME commands as in:

```
TIME 8:20
```

The new time must be entered using numerals only; letters are not allowed. The allowed options are:

<hh> = 00-24
<mm> = 00-59

The hour and minute entries must be separated by colons. You do not have to type the <ss> (seconds) or <cc> (hundredths of seconds) options.

MS-DOS uses the time entered as the new time if the options and separators are valid. If the options or separators are not valid, MS-DOS displays the message:

```
Invalid time
Enter new time:_
```

MS-DOS then waits for you to type a valid time.

TREE

Name: TREE

Type: External

Purpose: Displays all of the directory paths and lists the files in each sub-directory.

Syntax: [*drive*:] [*pathname*] TREE [*drive*:] [/F]

Comments: The *drive*: option specifies the drive. If omitted, the default drive currently assigned is assumed.

If each directory is found, its full pathname is displayed. The names of sub-directories each directory has will also be listed.

If the /F option is specified, names of files involved in each sub-directories will be shown.

An example:

```
TREE A:/F>TEST1.LST
```

saves all sub-directories and files to be listed with filename TEST1.LST.

TYPE

Name: TYPE

Type: Internal

Purpose: Displays the contents of the file on the console screen.

Syntax: TYPE <*filespec*>

Comments: Use this command to examine a file without modifying it. (Use DIR to find the name of a file and EDLIN to alter the contents of a file.) The only formatting performed by TYPE is that tabs are expanded to spaces consistent with tab stops every eighth column. Note that a display of binary files causes control characters (such as CONTROL-Z) to be sent to your computer, including bells, form feeds, and escape sequences.

VER

Name: VER

Type: Internal

Purpose: Prints MS-DOS version number.

Syntax: VER

Comments: If you want to know that version of MS-DOS you are using, type VER. The version number will be displayed on your screen.

VERIFY

Name: VERIFY

Type: Internal

Purpose: Turns the verify switch on or off when writing to disk.

Syntax: VERIFY [ON|OFF]

Comments: This command has the same purpose as the /V switch in the COPY command. If you want to verify that all files are written correctly to disk, you can use the VERIFY command to tell MS-DOS to verify that your files are intact (no bad sectors, for example). MS-DOS will perform a VERIFY each time you write data to a disk. You will receive an error message only if MS-DOS was unable to successfully write your data to disk.

VERIFY ON remains in effect until you change it in a program (by a SET VERIFY system call), or until you issue a VERIFY OFF command to MS-DOS.

If you want to know that the current setting of VERIFY is, type VERIFY with no options.

VOL

Name: VOL (VOLUME)

Type: Internal

Purpose: Displays disk volume label, if it exists.

Syntax: VOL [*drive*:]

Comments: This command prints the volume label of the disk in drive d:. If no drive is specified, MS-DOS prints the volume label of the disk in the default drive.

If the disk does not have a volume label, VOL displays:

Volume in drive x has no label

5.4 BATCH PROCESSING COMMANDS

The following commands are called batch processing commands. They can add flexibility and power to your batch programs. The commands discussed are ECHO, FOR, GOTO, IF and SHIFT.

If you are not writing batch programs, you do not need to read this section.

ECHO

Name: ECHO

Type: Internal

Purpose: Turns batch echo feature on and off

Syntax: ECHO [ON | OFF| *message*]

Comments: Normally, commands in a batch file are displayed ("echoed") on the console when they are seen by the command processor. ECHO OFF turns off this feature. ECHO ON turns the echo back on.

If ON or OFF are not specified, the current setting is displayed.

FOR

Name: FOR

Type: Internal

Purpose: Command extension used in batch and interactive file processing.

Syntax: FOR %%<*c*> IN <*set*> DO <*command*> - for batch processing
FOR %<*c*> IN <*set*> DO <*command*> - for interactive processing

Comments: <*c*> can be any character except 0,1,2,3,..,9 to avoid confusion with the %0-%9 batch parameters.

<*set*> is (# <*item*> * #)

The %%<*c*> variable is set sequentially to each member of <*set*>, and then <*command*> is evaluated. If a member of <*set*> is an expression involving *and/or ?, then the variable is set to each matching pattern from disk. In this case, only one such <*item*> may be in the set, and any <*item*> besides the first is ignored.

Examples:

```
FOR %%f IN (*.ASM) DO MASM %%f;

For %%f IN (FOO BAR BLECH) DO REM
%%f
```

The '%%' is needed so that after batch parameter (%0-%9) processing is done, there is one '%' left. If only '%f' were there, the batch parameter processor would see the '%', look at 'f', decide that '%f' was an error (bad parameter reference) and throw out the '%f', so that the command FOR would never see it. If the FOR is not in a batch file, then only one '%' should be used.

GOTO

Name: GOTO

Type: Internal

Purpose: Command extension used in batch file processing.

Syntax: GOTO *<label>*

Comments: GOTO causes commands to be taken from the batch file beginning with the line after the *<label>* definition. If no label has been defined, the current batch file will terminate.

Example:

```
:foo
REM looping...
GOTO foo
```

will produce an infinite sequence of messages: REM looping.....

Starting a line in a batch file with ':' causes the line to be ignored by batch processing. The characters following GOTO define a label, but this procedure may also be used to put in comment lines.

IF

Name: IF

Type: Internal

Purpose: Command extension used in batch file processing.

Syntax: IF *<condition>* *<command>*

Comments: The parameter *<condition>* is one of the following:

ERRORLEVEL *<number>*
: True if and only if the previous program executed by COMMAND had an exit code of *<number>* or higher.

<string1> = *<string2>*
: True if and only if *<string1>* and *<string2>* are identical after parameter substitution. Strings may not have embedded separators.

EXIST *<filename>*
: True if and only if *<filename>* exists.

NOT *<condition>*
: True if and only if *<condition>* is false.

The IF statement allows conditional execution of commands. When the *<condition>* is true, then the *<command >* is executed. Otherwise, the *<command>* is ignored.

Example:

```
IF NOT EXIST \TMP\FOO ECHO Can't find file
```

PAUSE

Name: PAUSE

Type: Internal

Purpose: Suspends execution of the batch file.

Syntax: PAUSE [*comment*]

Comments: During the execution of a batch file, you may need to change disks or perform some other action. PAUSE suspends execution until you press any key, except <CONTROL-C>.

When the command processor encounters PAUSE, it prints:

```
Strike a key when ready...
```

If you press <CONTROL-C>, another prompt will be displayed:

```
Terminate batch job (Y/N)?
```

If you type Y in response to this prompt, execution of the remainder of the batch command file will be aborted and control will be returned to the operating system command level. Therefore, PAUSE can be used to break a batch file into pieces, allowing you to end the batch command file at an intermediate point.

The comment is optional and may be entered on the same line as PAUSE. You may also want to prompt the user of the batch file with some meaningful message when the batch file pauses. For example, you may want to change disks in one of the drives. An optional prompt message may be given in such cases. The comment prompt will be displayed *before* the "Strike a key" message.

SHIFT

Name: SHIFT

Type: Internal

Purpose: Allows access to more than 10 replaceable parameters in batch file processing.

Syntax: SHIFT

Comments: Usually, command files are limited to handling 10 parameters, %0 through %9. To allow access to more than ten parameters, use SHIFT to change the command line parameters. For example:

If %0 = "foo"
%1 = "bar"
%2 = "name"
%3 ... %9 are empty

then a SHIFT will result in the following:

%0 = "bar"
%1 = "name"
%2...%9 are empty

If there are more than 10 parameters given on a command line, those that appear after the 10th (%9) will be shifted one at a time into %9 by successive shifts.

CHAPTER 6

MS-DOS EDITING AND FUNCTION KEYS

Special MS-DOS Editing Keys
Control Character Functions

6.1 SPECIAL MS-DOS EDITING KEYS

The special editing keys deserve particular emphasis because they depart from the way in which most operating systems handle command input. You do not have to type the same sequences of keys repeatedly, because the last command line is automatically placed in a special storage area called a template.

By using the template and the special editing keys, you can take ad vantage of the following MS-DOS features:

1. A command line can be instantly repeated by pressing two keys.

2. If you make a mistake in the command line, you can edit it and retry without having to retype the entire command line.

3. A command line that is similar to a preceding command line can be edited and executed with a minimum of typing by pressing a special editing key.

The relationship between the command line and the template is shown in Figure 8.

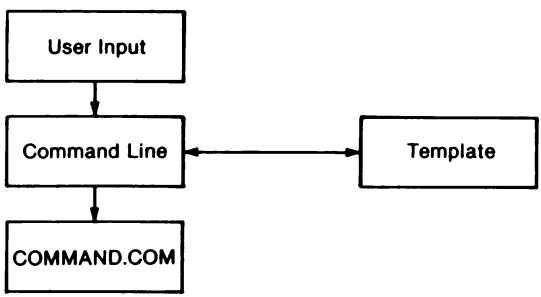


Figure 8. Command Line and Template

As seen in Figure 8, the user (you) types a command to MS-DOS on the command line. When you press the <ENTER> key, the command is automatically sent to the command processor (COMMAND.COM) for execution. At the same time, a copy of this command is sent to the template. You can now recall the command or modify it with MS-DOS special editing key.

Table 6.1 contains a complete list of the special editing keys. Each of these keys is more fully described in Chapter 7, The Line Editor (EDLIN), where they can be used to edit your text files.

Table 6.1 Special Editing Functions

Key	Editing Function	Your Keyboard
<COPY1>	Copies one character from the template to the command line.	F1
<COPYUP>	Copies characters up to the character specified in the template and puts these characters on the command line.	F2
<COPYALL>	Copies all remaining characters in the template to the command line.	F3
<SKIP1>	Skips over (does not copy) a character in the template.	DEL
<SKIPUP>	Skips over (does not copy) the characters in the template up to the character specified.	F4
<VOID>	Voids the current input; leaves the template unchanged.	ESC
<INSERT>	Enters/exits insert mode.	INS (toggle)
<NEWLINE>	Makes the new line in the new template.	F5
<CONTROL-Z>	Puts a CONTROL-Z (1AH) end-of-file character in the new template.	F6

Examples:

If you type the following command

DIR PROG.COM

MS-DOS displays information about the file PROG.COM on your screen. The command line is also saved in the template. To repeat the command, just press two keys: <COPYALL> (F3) and <ENTER>.

The repeated command is displayed on the screen as you type, as shown below:

<COPYALL> DIR PROG.COM <ENTER>

Notice that pressing the <COPYALL> (F3) key causes the contents of the template to be copied to the command line; pressing <ENTER> causes the command line to be sent to the command processor for execution.

If you want to display information about a file named PROG.ASM, you can use the contents of the template and type:

<F2> C

Typing <F2> C copies all characters from the template to the command line, up to but not including C.MS-DOS displays:

DIR PROG._

Note that the underline is your cursor. Now type:

ASM

The result is:

DIR PROG.ASM_

The command line DIR PROG.ASM is now in the template and ready to be sent to the command processor for execution. To do this, press <ENTER>.

Now assume that you want to execute the following command:

TYPE PROG.ASM

To do this, type:

TYPE<INS> <F3><ENTER>

Notice that when you are typing, the characters are entered directly into the command line and overwrite corresponding characters in the template. This automatic replacement feature is turned off when you press the insert key. Thus,the characters "TYPE" replace the characters "DIR" in the template. To insert a space between "TYPE" and "PROG.ASM", you pressed <INS> and then the space bar. Finally, to copy the rest of the template to the command line, you pressed <F3> and then <ENTER>. The command TYPE PROG.ASM has been processed by MS-DOS, and the template becomes TYPE PROG.ASM.

If you had misspelled TYPE as BYTE, a command error would have occurred. Still, instead of throwing away the whole command, you could save the misspelled line before you press <ENTER> by creating a new template with the <F5> key:

BYTE PROG.ASM<F5>

You could then edit this erroneous command by typing:

T<F1>P<F3>

The<F1> key copies a single character from the template to the command line. The resulting command line is then the command that you want:

TYPE PROG.ASM

As an alternative, you can use the same template containing BYTE PROG.ASM and then use the <DEL> and <INS>keys to achieve the same result:

<DEL><DEL><F1><INS>YP<F3>

To illustrate how the command line is affected as you type, examine the keys typed on the left; their effect on the command line is shown on the right:

<DEL>	_	Skips over 1st template character
<DEL>	_	Skips over 2nd template character
<F1>	T	Copies 3rd template character
<INS>YP	TYP	Inserts two characters
<F3>	TYPE PROG.ASM	Copies rest of template

Notice that <DEL> does not affect the command line. It affects the template by deleting the first character. Similarly, <F4> deletes characters in the template, up to but not including a given character.

These special editing keys can add to your effectiveness at the keyboard. The next section describes control character functions that can also help when you are typing commands.

6.2 CONTROL CHARACTER FUNCTIONS

A control character function is a function that affects the command line. You have already learned about <CONTORL-C> and <CONTROL-S>. Other control character functions are described below.

Remember that when you type a control character, such as <CONTROL-C>, you must hold down the control key and then press the C key.

Table 6.2 Control Character Functions

Control Character	Function
<CONTROL-BREAK>	Aborts current command.
<CONTROL-H>	Removes last character from command line, and erases character from terminal screen.
<CONTROL-I>	Moves the cursor to the next horizontal tabulation position.
<CONTROL-J>	Inserts physical end-of-line, but does not empty command line. Use the <LINE FEED> key to extend the current logical line beyond the physical limits of one terminal screen.
<CONTROL-P>	Toggles terminal output to line printer.
<CONTROL-S>	Suspends output display on terminal screen. Press any key to resume.
<CONTROL-U>	Deletes the current line on the display.
<CONTROL-[>	Cancels the current line; empties the command line; and then outputs a back slash(\ ), carriage return, and line feed. The template used by the special editing commands is not affected.

CHAPTER 7

THE LINE EDITOR(EDLIN)

Introduction
How to Start EDLIN
Special Editing Keys
Command Information
  Command Options
EDLIN Commands
Error Messages

7.1 INTRODUCTION

In this chapter, you will learn how to use EDLIN, the line editor program. You can use EDLIN to create, change, and display files, whether they are source program or text files.

You can use EDLIN to:

1. Create new source files and save them.

2. Update existing files and save both the updated and original files.

3. Delete, edit, insert, and display lines.

4. Search for, delete, or replace text within one or more lines.

The text in files created or edited by EDLIN is divided into lines, each up to 253 characters long. Line numbers are generated and displayed by EDLIN during the editing process, but are not actually present in the saved file.

When you insert lines, all line numbers following the inserted text advance automatically by the number of lines being inserted. When you delete lines in a file, all line numbers following the deleted text decrease automatically by the number of lines deleted. As a result, lines are always numbered consecutively in your file.

7.2 HOW TO START EDLIN

To start EDLIN, type:

EDLIN <*filespec*>

If you are creating a new file, the <filespec> should be the name of the file you wish to create. If EDLIN does not find this file on a drive, EDLIN will create a new file with the name you specify. The following message and prompt will be displayed:

```
New file
*_
```

Notice that the prompt for EDLIN is an asterisk (*)

You can now type lines of text into your new file. To begin entering text, you must enter an I (Insert) command to insert lines. The I command is discussed later in this chapter.

If you want to edit an existing file, <*filespec*> should be the name of the file you want to edit. When EDLIN finds the file you specify on the designated or default drive, the file will be loaded into memory. If the entire file can be loaded, EDLIN will display the following message on your screen:

```
End of input file
*
```

You can then edit the file using EDLIN editing commands.

If the file is too large to be loaded into memory, EDLIN will load lines until memory is 3/4 full, then display the * prompt. You can then edit the portion of the file that is in memory.

To edit the remainder of the file, you must save some of the edited lines on disk to free memory; then EDLIN can load the unedited lines from disk into memory. Refer to the Write and Append commands in this chapter for the procedure.

When you complete the editing session, you can save the original and the updated (new) files by using the End command. The End command is discussed in this chapter in the section "EDLIN Commands". The original file is renamed with an extension of .BAK, and the new file has the filename and extension you specify in the EDLIN command. The original .BAK file will not be erased until the end of the editing session, or until disk space is needed by the editor (EDLIN).

Do not try to edit a file with a filename extension of .BAK because EDLIN assumes that any .BAK file is a backup file. If you find it necessary to edit such a file, rename the file with another extension (using the MS-DOS RENAME command discussed in Chapter 5), then start EDLIN and specify the new <*filespec*>.

7.3 SPECIAL EDITING KEYS

The special editing keys and template discussed in Chapter 6 can be used to edit your text files. These keys are discussed in detail in this section.

Table 7.1 summarizes the commands, codes, and functions. Descriptions of the special editing keys follow the table.

Table 7.1 Special Editing Keys

Function	Key	Description
Copy one character	F1	Copies one character from the template to the new line.
Copy up to character	F2	Copies all characters from the template to the new line, up to the character specified.
Copy template	F3	Copies all remaining characters in the template to the screen.
Skip one character	DEL	Does not copy (skips over) a character.
Skip up to character	F4	Does not copy (skips over) the characters in the template, up to the character specified.
Quit input	ESC	Voids the current input; leaves the template unchanged.
Insert mode	INS	Enters/exits insert mode.
Replace mode	F6	Turns insert mode off; this is the default.
New template	F5	Makes the new line in the new template.

<F1>

Key: <F1>

Purpose: Copies one character from the template to the command line.

Comments: Pressing the <F1> key copies one character from the template to the command line. When the <F1> key is pressed, one character is inserted in the command line and insert mode is automatically turned off.

Example: Assume that the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Pressing the <F1> key copies the first character (T) to the second of the two lines displayed:

```
                1:*This is a sample file
<F1>            1:*T_
```

Each time the <F1> key is pressed, one more character appears:

```
<F1>    1:*Th_
<F1>    1:*Thi_
<F1>    1:*This_
```

<F2>

Key: <F2>

Purpose: Copies multiple characters up to a given character.

Comments: Pressing the <F2> key copies all characters up to a given character from the template to the command line. The given character is the next character typed after <F2>; it is not copied or displayed on the screen. Pressing the <F2> key causes the cursor to move to the single character that is specified in the command. If the template does not contain the specified character, nothing is copied. Pressing <F2> also automatically turns off insert mode.

Example: Assume that the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Pressing the <F2> key copies all characters up to the character specified immediately after the <F2> key.

```
                    1:*This is a sample file
<F2>p               1:*This is a sam_
```

<F3>

Key: <F3>

Purpose: Copies template to command line.

Comments: Pressing the <F3> key copies all remaining characters from the template to the command line. Regardless of the cursor position at the time the <F3> key is pressed, the rest of the line appears, and the cursor is positioned after the last character on the line.

Example: Assume that the screen shows:

```
1:*This is a sample file
1:*__
```

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Pressing the <F3> key copies all characters from the template (shown in the upper line displayed) to the line with the cursor (the lower line displayed):

```
                1:*This is a sample file (template)
<F3>            1:*This ia a sample file.__ (command line)
```

Also, insert mode is automatically turned off.

DEL

Key: <DEL>

Purpose: Skips over one character in the template.

Comments: Pressing the <DEL> key skips over one character in the template. Each time you press the <DEL> key, one character is not copied from the template. The action of the <DEL> key is similar to the <F1> key, except that <DEL> skips a character in the template rather than copying it to the command line.

Example: Assume that the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Pressing the <DEL> key skips over the first character (T).

```
          1:*This ia a sample file
<DEL>  1:*_
```

The cursor position does not change and only the template is affected. To see how much of the line has been skipped over, press the <F3> key, which moves the cursor beyond the last character of the line.

```
          1:*This is a sample file.
<DEL>  1:*_
<F3>   1:*his is a sample file.—
```

<F4>

Key: <F4>

Purpose: Skips multiple characters in the template up to the specified character.

Comments: Pressing the <F4> key skips over all characters up to a given character in the template. This character is not copied and is not shown on the screen. If the template does not contain the specified character, nothing is skipped over. The action of the <F4> key is similar to the <COPYUP> key, except that <F4> skips over characters in the template rather than copying them to the command line.

Example: Assume that the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Pressing the <F4> key skips over all the characters in the template up to the character pressed after the <F4> key:

```
                1:*This is a sample file
<F4>p 1:*_
```

The cursor position does not change. To see how much of the line has been skipped over, press the <F3>key to copy the template. This moves the cursor beyond the last character of the line:

```
                1:*This is a sample file:
<F4>p 1:*_
<F3>  1:*ple file._
```

<ESC>

Key: <ESC>

Purpose: Quits input and empties the command line.

Comments: Pressing the <ESC> key empties the command line, but it leaves the template unchanged. <ESC> also prints a back slash (\), carriage return, and line feed, and turns insert mode off. The cursor (indicated by the underline) is positioned at the beginning of the line. Pressing the <F3> key copies the template to the command line and the command line appears as it was before <ESC> was pressed.

Example: Assume that the screen shows:

```
1:*This is a sample file.
2:*_
```

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Assume that you want to replace the line with "Sample File:"

```
1:*This is a sample file.
2:*Sample File_
```

To cancel the line you just entered (Sample File), and to keep "This is a sample file.", press <ESC>. Notice that a backslash appears on the Sample File line to tell you it has been cancelled.

```
                1:*This is a sample file.
<ESC>           1:*Sample File\
                _
```

Press <ENTER> to keep the original line, or to perform any other editing functions. If <F3> is pressed, the original template is copied to the command line:

```
<F3>        This is a sample file._
```

<INS>

Key: <INS>

Purpose: Enters/exits insert mode.

Comments: Pressing the <INS> key causes EDLIN to enter and exit insert mode. The current cursor position in the template is not changed. The cursor does move as each character is inserted. However, when you have finished inserting characters, the cursor will be positioned at the same character as it was before the insertion began. Thus, characters are inserted in front of the character to which the cursor points.

Example: Assume that the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the editing session,the cursor (indicated by the underline) is positioned at the beginning of the line. Assume that you press the <F2> and f keys:

```
                1:*This is a sample file
<F2>f           2:*This is a sample_
```

Now press the <INS> key and insert the characters "edit" and a space:

```
                1:*This is a sample file.
<F2>f           2:*This is a sample_
<INS>edit       2:*This is a sample edit_
```

If you now press the <F3> key, the rest of the template is copied to the line:

```
                1:*This is a sample edit_
<F3>            2:*This is a sample edit file._
```

If you pressed the <ENTER> key, the remainder of the template would be truncated, and the command line would end at the end of the insert:

```
<INS> edit <ENTER>
1:*This is a sample edit_
```

To exit insert mode, simply press the <INS> key again.

<F6>

Key: <F6>

Purpose: Enters replace mode.

Comments: Pressing the <F6> key causes EDLIN to exit insert mode and to enter replace mode. All the characters you type will overstrike and replace characters in the template. When you start to edit a line, replace mode is in effect. If the <ENTER> key is pressed, the remainder of the template will be deleted.

Example: Assume that the screen shows:

```
1:*This is a sample file.
2:*_
```

At the beginning of the editing session, the cursor (indicated by the underline) is positioned at the beginning of the line. Assume that you then press <F2>m, <INS>lary, <F6>tax, and then <F3>:

```
                1:*This is a sample file.
<F2>m           2:*This is a sa_
<INS>lary       2:*This is a salary_
<F6>tax         2:*This is a salary tax_
<F3>            2:*This is a salary tax file._
```

Notice that you inserted lary and replaced mple with tax. If you type characters that extend beyond the length of the template, the remaining characters in the template will be automatically appended when you press <F3>.

<F5>

Key: <F5>

Purpose: Creates a new template.

Comments: Pressing the <F5> key copies the current command line to the template. The contents of the old template are deleted. Pressing <F5> outputs an @ ("at sign" character), a carriage return, and a line feed. The command line is also emptied and insert mode is turned off.

NOTE

<F5> performs the same function as the <ESC> key, except that the template is changed and an @ ("at sign" character) is printed instead of a \ (backslash).

Example: Assume that the screen shows:

```
1:*This is a sample file.
2:*_
```

At the beginning of the editing session,the cursor (indicated by the underline) is positioned at the beginning of the line. Assume that you enter <F2>m, <INS>lary, <F6>tax, and then <F3>:

```
                1:*This is a sample file.
<F2>m           2:*This is a sa_
<INS>lary       2:*This is a salary_
<F6>tax         2:*This is a salary tax_
<F3>            2:*This is a salary tax file._
```

At this point, assume that you want this line to be the new template, so you press the <F5> key:

<F5> **1:*This is a salary tax file.@**

The @ indicates that this new line is now the new template. Additional editing can be done using the new template.

7.4 COMMAND INFORMATION

EDLIN commands perform editing functions on lines of text. The following list contains information you should read before you use EDLIN commands.

1. Pathnames are acceptable as options to commands. For example, typing EDLIN \BIN\USER\JOE\TEXT.TXT will allow you to edit the TEXT.TXT file in the subdirectory JOE.

2. You can reference line numbers relative to the current line (the line with the asterisk). Use a minus sign with a number to indicate lines before the current line. Use a plus sign with a number to indicate lines after the current line.

   Example:

   ```
   -10, +10L
   ```

   This command lists 10 lines before the current line, the current line, and 10 lines after the current line.

3. Multiple commands may be issued on one command line. When you issue a command to edit a single line using a line number (<line>), a semicolon must separate commands on the line. Otherwise, one command may follow another without any special separators. In the case of a Search or Replace command, the <string> may be ended by a <CONTROL-Z> instead of a <ENTER>.

   Examples:

   The following command line edits line 15 and then displays lines 10 through 20 on the screen.

   ```
   15;-45, +5L
   ```

   The command line in the next example searches for "This string" and then displays 5 lines before and 5 lines after the line containing the matched string. If the search fails, then the displayed lines are those line numbers relative to the current line.

   ```
   SThis string <CONTROL-Z> -5, +5L
   ```

4. You can type EDLIN commands *with* or *without* a space between the line number and command. For example, to delete line 6, the command 6D is the same as 6 D.

5. It is possible to insert a control character (such as CONTROL-C) into text by using the quote character CONTROL-V before it while in insert mode. CONTROL-V tells MS-DOS to recognize the next *capital* letter typed as a control character. It is also possible to use a control character in any of the string arguments of Search or Replace by using the special quote character. For example:

   S<CONTROL-V>Z
   will find the first occurrence
   of CONTROL-Z in a file

   R<CONTROL-V>Z<CONTROL-Z>foo
   will replace all occurrences
   of CONTROL-Z in a file by foo

   S<CONTROL-V>C<CONTROL-Z>bar
   will replace all occurrences of
   CONTROL-C by bar

   It is possible to insert CONTROL-V into the text by typing CONTROL-V-V.

6. The CONTROL-Z character ordinarily tells EDLIN, "This is the end of the file." If you have CONTROL-Z characters elsewhere in your file, you must tell EDLIN that these other control characters do not mean end-of-file. Use the /B switch to tell EDLIN to ignore any CONTROL-Z characters in the file and to show you the entire file.

The EDLIN commands are summarized in the following table. They are also described in further detail following the description of command options.

Table 7.2 EDLIN Commands

Command	Purpose
<line>	Edits line no.
A	Appends lines
C	Copies lines
D	Deletes lines
E	Ends editing
I	Inserts lines
L	Lists text
M	Moves lines
P	Pages text
Q	Quits editing
R	Replaces lines
S	Searches text
T	Transfers text
W	Writes lines

7.4.1 Command Options

Several EDLIN commands accept one or more options. The effect of a command option varies, depending on with which command it is used. The following list describes each option.

<line> <line> indicates a line number that you type. Line numbers must be separated by a comma or a space from other line numbers, other options, and from the command.

<line> may be specified one of three ways:

Number Any number less than 65534. If a number larger than the largest existing line number is specified, then <line> means the line after the last line number.

Period (.) If a period is specified for <line>, then <line> means the current line number. The current line is the last line edited, and is not necessarily the last line displayed. The current line is marked on your screen by an asterisk (*) between the line number and the first character.

Pound (#) The pound sign indicates the line after the last line number.
If you specify # for <line>, this has the same effect as specifying a number larger than the last line number.

<ENTER> A carriage return entered without any of the <line> specifiers listed above directs EDLIN to use a default value appropriate to the command.

The question mark option directs EDLIN to ask you if the correct string has been found. The question mark is used only with the Replace and Search commands. Before continuing, EDLIN waits for either a Y or <ENTER> for a yes response, or for any other key for a no response.

<string> <string> represents text to be found, to be replaced, or to replace other text. The <string> option is used only with the Search and Replace commands. Each <string> must be ended by a <CONTROL-Z> or a <ENTER> (see the Replace command for details). No space should be left between strings or between a string and its command letter, unless you want those spaces to be part of the string.

7.5 EDLIN COMMANDS

The following pages describe EDLIN editing commands.

APPEND

Name: Append

Purpose: Adds the specified number of lines from disk to the file being edited in memory. The lines are added at the end of lines that are currently in memory.

Syntax: [<*n*>] A

Comments: This command is meaningful only if the file being edited is too large to fit into memory. As many lines as possible are read into memory for editing when you start EDLIN.

To edit the remainder of the file that will not fit into memory, lines that have already been edited must be written to disk. Then you can load unedited lines from disk into memory with the Append command. Refer to the Write command in this chapter for information on how to write edited lines to disk.

NOTES

1. If you do not specify the number of lines to append, lines will be appended to memory until available memory is 3/4 full. No action will be taken if available memory is already 3/4 full.

2. the message "End of input file" is displayed when the Append command has read the last line of the file into memory.

COPY

Name: Copy

Purpose: Copies a range of lines to a specified line number. The lines can be copied as many times as you want by using the <*count*> option.

Syntax: [<*line*>],[<*line*>],<*line*>,[<*count*>] C

Comments: If you do not specify a number in <*count*>, EDLIN copies the lines one time. If the first of the second <*line*> are omitted, the default is the current line. The file is renumbered automatically after the copy.

The line numbers must not overlap or you will get an "Entry error" message. For example, 3, 20, 15C would result in an error message.

Examples: Assume that the following file exists and is ready to edit:

```
1:*This is a sample file
2: used to show copying lines.
3: See what happens when you use
4: the Copy command
5: (the C command)
6: to copy text in your file.
```

You can copy this entire block of text by issuing the following command:

```
1,6,7C
```

The result is:

```
1: This is a sample file
2: used to show copying lines.
3: See what happens when you use
4: the Copy command
5: (the C command)
6: to copy text in your file.
```

```
 7:*This is a sample file
 8: used to show copying lines.
 9: See what happens when you use
10: the Copy command
11: (the C command)
12: to copy text in your file.
```

If you want to place the text within other text, the third *<line>* option should specify the line *before* which you want the copied text to appear. For example, assume that you want to copy lines and insert them *within* the following file:

```
 1:*This is a sample file
 2: used to show copying lines.
 3: See what happens when you use
 4: the Copy command
 5: (the C command)
 6: to copy text in your file.
 7: You can also use COPY
 8: to copy lines of text
 9: to the middle of your file.
10: End of sample file.
```

The command 3, 6, 9C results in the following file:

```
 1: This is a sample file
 2: used to show copying lines.
 3: See what happens when you use
 4: the Copy command
 5: (the C command)
 6: to copy text in your file.
 7: You can also use COPY
 8: to copy lines of text
 9: to the middle of your file.
10:*See what happens when you use
11: the Copy command
12: (the C command)
13: to copy text in your file.
14: End of sample file.
```

DELETE

Name: Delete

Purpose: Deletes a specified range of lines in a file.

Syntax: [<*line*>] [,<*line*>] D

Comments: If the first <*line*> is omitted, that option will default to the current line (the line with the asterisk next to the line number). If the second <*line*> is omitted, then just the first <*line*> will be deleted. When lines have been deleted, the line immediately after the deleted section becomes the current line and has the same line number as the first deleted <*line*> had before the deletion occurred.

Examples: Assume that the following file exists and is ready to edit:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
.
.
.
25: (the D and I commands)
26: to edit the text
27:*in your file.
```

To delete multiple lines, type <*line*>,<*line*>D:

```
5,24D
```

The result is:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5:*(the D and I commands)
6: to edit text
7: in your file.
```

To delete a single line, type:

```
6D
```

The result is:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6:*in your file.
```

Next, delete a range of lines from the following file:

```
1: This is a sample file
2: used to show dynamic line numbers.
3:*See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7: in your file.
```

To delete a range of lines beginning with the current line, type:

```
,6D
```

The result is:

```
1: This is a sample file
2: used to show dynamic line numbers.
3:*in your file.
```

Notice that the lines are automatically renumbered.

EDIT

Name: Edit

Purpose: Edits line of text.

Syntax: [<*line*>]

Comments: When a line number is typed, EDLIN displays the line number and text; then, on the line below, EDLIN reprints the line number. The line is now ready for editing. You may use any of the EDLIN editing commands to edit the line. The existing text of the line serves as the template until the <ENTER> key is pressed.

If no line number is typed (that is, if only the <ENTER> key is pressed), the line after the current line (marked with an asterisk (*)) is edited. If no changes to the current line are needed and the cursor is at the beginning or end of the line, press the <ENTER> key to accept the line as it is.

WARNING

If the <ENTER> key is pressed while the cursor is in the middle of the line, the remainder of the line is deleted.

Example: Assume that the following file exists and is ready to edit:

```
1: This is a sample file.
2: used to show
3: the editing of line
4:*four.
```

To edit line 4, type:

```
4
```

The contents of the line are displayed with a cursor below the line:

```
4:*four.
4:*_
```

Now, using the <F3> key, type:

<INS> **number**
<F3><ENTER> **4: number four.**

END

Name: End

Purpose: Ends the editing session.

Syntax: E

Comments: This command saves the edited file on disk, renames the original input file <*filename*>.BAK, and then exits EDLIN. If the file was created during the editing session, no .BAK file is created.

The E command takes no options. Therefore, you cannot tell EDLIN on which drive to save the file. The drive you want to save the file on must be selected when the editing session is started. If the drive is not selected when EDLIN is started, the file will be saved on the disk in the default drive. It will still be possible to COPY the file to a different drive using the MS-DOS COPY command.

You must be sure that the disk contains enough free space for the entire file. If the disk does not contain enough free space, the write will be aborted and the edited file lost, although part of the file might be written out to the disk.

Example: E <ENTER>

After execution of the E command, the MS-DOS default drive prompt (for example, A>) is displayed.

INSERT

Name: Insert

Purpose: Inserts text immediately before the specified *<line>*.

Syntax: [*<line>*]I

Comments: If you are creating a new file, the I command must be given before text can be typed (inserted). Text begins with line number 1. Successive line numbers appear automatically each time <ENTER> is pressed.

EDLIN remains in insert mode until <CONTROL-C> is typed. When the insert is completed and insert mode has been exited, the line immediately following the inserted lines becomes the current line. All line numbers following the inserted section are incremented by the number of lines inserted.

If *<line>* is not specified, the default will be the current line number and the lines will be inserted immediately before the current line. If *<line>* is any number larger than the last line number, or if a pound sign (#) is specified as *<line>*, the inserted lines will be appended to the end of the file. In this case, the last line inserted will become the current line.

Examples: Assume that the following file exists and is ready to edit:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7:*in your file.
```

To insert text before a specific line that is not the current line, type <*line*> I:

```
7I
```

The result is:

```
7:_
```

Now, type the new text for line 7:

```
7:  and renumber lines
```

Then to end the insertion, press <CONTROL-Z> on the *next* line:

```
8: <CONTROL-Z>
```

Now type L to list the file. The result is:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7: and renumber lines
8:*in your file.
```

To insert lines immediately before the current line, type:

```
I
```

The result is:

```
8:_
```

Now, insert the following text and terminate with a <CONTROL-Z> on the next line:

```
8: so they are consecutive
9: <CONTROL-Z>
```

Now to list the file and see the result, type L:

The result is:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7: and renumber lines
8: so they are consecutive
9:*in your file.
```

To append new lines to the end of the file, type:

```
10I
```

This produces the following:

```
10:_
```

Now, type the following new lines:

```
10: The insert command can place new lines
11: in the file; there's no problem
12: because the line numbers are dynamic;
13: they'll go all the way to 65533.
```

End the insertion by pressing <CONTROL-Z> on line 14. The new lines will appear at the end of all previous lines in the file. Now type the List command, L:

The result is:

```
 1: This is a sample file
 2: used to show dynamic line numbers.
 3: See what happens when you use
 4: Delete and Insert
 5: (the D and I commands)
 6: to edit text
 7: and renumber lines
 8: so they are consecutive
 9: in your file.
10: The insert command can place new lines
11: in the file; there's no problem
12: because the line numbers are dynamic;
13: they'll go all the way to 65533.
```

LIST

Name: List

Purpose: Lists a range of lines, including the two lines specified.

Syntax: [<*line*>][,<*line*>]L

Comments: Default values are provided if either one or both of the options are omitted. If you omit the first option, as in:

,<*line*>L

the display will start 11 lines before the current line and end with the specified <*line*>. The beginning comma is required to indicate the omitted first option.

NOTE

If the specified <*line*> is more than 11 lines before the current line, the display will be the same as if you omitted both options.

If you omit the second option, as in

<*line*>L

23 lines will be displayed, starting with the specified <*line*>.

If you omit both parameters, as in

L

23 lines will be displayed — the 11 lines before the current line, the current line, and the 11 lines after the current line. If there are less than 11 lines before the current line, more than 11 lines after the current line will be displayed to make a total of 23 lines.

Examples: Assume that the following file exists and is ready to edit:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
.
.
15:*The current line contains an asterisk.
.
.
.
26: to edit text
27: in your file.
```

To list a range of lines without reference to the current line, type <*line*>, <*line*> L:

```
2,5L
```

The result is:

```
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
```

To list a range of lines beginning with the current line, type, <*line*> L:

```
,26L
```

The result is:

```
15:*The current line contains an asterisk.
.
.
.
26 to edit text
```

To list a range of 23 line centered around the current line, type only L:

```
L
```

The result is:

```
 4: Delete and Insert
 5: (the D and I commands)
 .
 .
 .
13: The current line is listed in the middle.
14: The current line remains unchanged.
15:*The current line contains an asterisk.
 .
 .
 .
26: to edit text.
```

MOVE

Name: Move

Purpose: Moves a range of text to the line specified.

Syntax: [<*line*>], [<*line*>], <*line*> **M**

Comments: Use the Move command to move a block of text (from the first <*line*> to the second <*line*>) to another location in the file. The lines are renumbered according to the direction of the move. For example,

```
,+25,100M
```

moves the text from the current line plus 25 lines to line 100. If the line numbers overlap, EDLIN will display an "Entry error" message.

To move lines 20-30 to line 100, type:

```
20,30,100M
```

PAGE

Name: Page

Purpose: Pages through a file 23 lines at a time.

Syntax: [<*line*>][,<*line*>]P

Comments: If the first <*line*> is omitted, that number will default to the current line plus one. If the second <*line*> is omitted, 23 lines will be listed. The new current line becomes the last line displayed and is marked with an asterisk.

QUIT

Name: Quit

Purpose: Quits the editing session, does *not* save any editing changes, and exits to the MS-DOS operating system.

Syntax: Q

Comments: EDLIN prompts you to make sure you don't want to save the changes.

Type Y if you want to quit the editing session. No editing changes are saved and no .BAK file is created. Refer to the End command in this chapter for information about the .BAK file.

Type N or any other character except Y if you want to continue the editing session.

NOTE

When started, EDLIN erases any previous copy of the file with an extension of .BAK to make room to save the new copy. If you reply Y to the Abort edit (Y/N)? message, your previous backup copy will no longer exist.

Example:

```
Q
Abort edit (Y/N)? Y <ENTER>
A>_
```

REPLACE

Name: Replace

Purpose: Replaces all occurrences of a string of text in the specified range with a different string of text or blanks.

Syntax: [<*line*>] [,<*line*>] [?]R<*string1*>
<CONTROL-Z><*string2*>

Comments: As each occurrence of <*string1*> is found, it is replaced by <*string2*>. Each line in which a replacement occurs will be displayed. If a line contains two or more replacements of <*string1*> with <*string2*>, then the line will be displayed once for each occurrence. When all occurrences of <*string1*> in the specified range are replaced by <*string2*>, the R command terminates and the asterisk prompt reappears.

If a second string is to be given as a replacement, then <*string1*> must be separated from <string2> with a <CONTROL-Z>. <*String2*> *must* also be ended with a <CONTROL-Z> <ENTER> combination or with a simple <ENTER>.

If <*string1*> is omitted, then Replace will take the old <*string1*> as its value, If there is no old <*string1*>, i.e., this is the first replace done, then the replacement process will be terminated immediately. If <*string2*> is omitted, then <*string1*> may be ended with a <ENTER>. If the first <*line*> is omitted in the range argument (as in,<*line*>) then the first <*line*> will default to the line after the current line. If the second <*line*> is omitted (as in <*line*> or <*line*>,), the second <*line*> will default to #. Therefore, this is the same as <*line*>,#. Remember that # indicates the line after the last line of the file.

If <*string1*> is ended with a <CONTROL-Z> and there is no <*string2*>, <*string2*> will be taken as an empty string and will become the new replace string. For example,

R <*string2*> <CONTROL-Z> <ENTER>

will delete occurrences of <*string1*>, but

R<*string1*> <return> and
R<ENTER>

will replace <*string1*> by the old <*string2*> and the old <*string1*> with the old <*string2*>, respectively. Note that "old" here refers to a previous string specified either in a Search or a Replace command.

If the question mark (?) option is given, the Replace command will stop at each line with a string that matches <*string1*>, display the line with <string2> in place, and then display the prompt O.K.?. If you press Y or the <ENTER> key, then <string2> will replace <*string1*>, and the next occurrence of <*string1*> will be found. Again, the O.K.? prompt will be displayed. This process will continue until the end of the range or until the end of the file. After the last occurrence of <*string1*> is found, EDLIN displays the asterisk prompt.

If you press any key besides Y or <ENTER> after the O.K.? prompt, the <*string1*> will be left as it was in the line, and Replace will go to the next occurrence of <*string1*>. If <*string1*> occurs more than once in a line, each occurrence of <*string1*> will be replaced individually, and the O.K.? prompt will be displayed after each replacement. In this way, only the desired <*string1*> will be replaced, and you can prevent unwanted substitutions.

Examples: Assume that the following file exists and is ready for editing:

```
 1: This is a sample file
 2: used to show dynamic line numbers.
 3: See what happens when you use
 4: Delete and Insert
 5: (the D and I commands)
 6: to edit text
 7: in your file.
 8: The insert command can place new lines
 9: in the file; there's no problem
10: because the line numbers are dynamic;
11: they'll go all the way to 65533.
```

To replace all occurrences of *<string1>* with *<string2>* in a specified range, type:

2,12 Rand <CONTROL-Z>or<ENTER>

The result is:

```
 4: Delete or Insert
 5: (the D or I commors)
 8: The insert commor can place new lines
```

Note that in the above replacement, some unwanted substitutions have occurred. To avoid these and to confirm each replacement, the same original file can be used with a slightly different command.

In the next example, to replace only certain occurrences of the first *<string>* with the second *<string>*, type:

2? Rand <CONTROL-Z> or <ENTER>

The result is:

```
4: Delete or Insert
O.K.? Y
5: (The D or I commands)
O.K.? Y
5: (The D or I commors)
O.K.? N
8: The insert commor can place new lines
O.K.? N
*_
```

Now, type the List command (L) to see the result of all these changes:

```
.
.
4: Delete or Insert
5: (The D or I commands)
.
8: The insert command can place new lines
.
.
```

SEARCH

Name: Search

Purpose: Searches the specified range of lines for a specified string of text.

Syntax: [<*line*>][,<*line*>][?]S<*string*><ENTER>

Comments: The <*string*> must be ended with a <ENTER>. The first line that matches <*string*> is displayed and becomes the current line. If the question mark option is not specified, the Search command will terminate when a match is found. If no line contains a match for <*string*>, the message "Not found" will be displayed.

If the question mark option (?) is included in the command, EDLIN will display the first line with a matching string; it will then prompt you with the message O.K.?. If you press either the Y or <ENTER> key, the line will become the current line and the search will terminate. If you press any other key, the search will continue until another match is found, or until all lines have been searched (and the Not found message is displayed).

If the first <*line*> is omitted (as in, <*line*> S <*string*>), the first <*line*> will default to the line after the current line. If the second <*line*> is omitted (as in <*line*> S <*string*> or <*line*>, S <*string*>), the second <*line*> will default to # (line after last line of file), which is the same as <*line*>, #S <*string*>. If <*string*> is omitted, Search will take the old string if there is one. (Note that "old" here refers to a string specified in a previous Search or Replace command.) If there is not an old string (i.e., no previous search or replace has been done), the command will terminate immediately.

Examples: Assume that the following file exists and is ready for editing:

```
1: This is a sample file
2: used to show dynamic line numbers.
3: See what happens when you use
4: Delete and Insert
5: (the D and I commands)
6: to edit text
7: in your file.
8: The insert command can place new lines
9: in the file; there's no problem
10: because the line numbers are dynamic;
11:*they'll go all the way to 65533.
```

To search for the first occurrence of the string "and", type

```
2,12 Sand <ENTER>
```

The following line is displayed:

```
4:Delete and Insert
```

To get the "and" in line 5, modify the search command by typing:

```
<DEL><F3>, 12 Sand<ENTER>
```

The search then continues from the line after the current line (line 4), since no first line was given. The result is:

```
5: (the D and I commands)
```

To search through several occurrences of a string until the correct string is found, type:

```
1,? Sand
```

The result is:

```
4: Delete and Insert
O,K.?_
```

If you press any key (except Y or <ENTER>), the search continues, so type N here:

```
O.K.? N
```

Continue:

```
5: (the D and I commands)
O.K.?_
```

Now press Y to terminate the search:

```
O.K.? Y
*_
```

To search for string XYZ without the verification (O.K.?), type:

```
SXYZ
```

EDLIN will report a match and will continue to search for the same string when you issue the S command:

```
S
```

EDLIN reports another match.

```
S
```

EDLIN reports the string is not found.

Note that <*string*> defaults to any string specified by a previous Replace or Search command.

TRANSFER

Name: Transfer

Purpose: Inserts (merges) the contents of <*filename*> into the file currently being edited at <*line*>. If <*line*> is omitted, then the current line will be used.

Syntax: [<*line*>] T <*filename*>

Comments: This command is useful if you want to put the contents of a file into another file or into the text you are typing. The transferred text is inserted at the line number specified by <*line*> and the lines are renumbered.

WRITE

Name: Write

Purpose: Writes a specified number of lines to disk from the lines that are being edited in memory. Lines are written to disk beginning with line number 1.

Syntax: [<*n*>] W

Comments: This command is meaningful only if the file you are editing is too large to fit into memory. When you start EDLIN, EDLIN reads lines into memory until memory is 3/4 full.

To edit the remainder of your file, you must write edited lines in memory to disk. Then you can load additional unedited lines from disk into memory by using the Append command.

NOTE

If you do not specify the number of lines, lines will be written until memory is 3/4 full. No action will be taken if available memory is already more than 3/4 full. All lines are renumbered, so that the first remaining line becomes line number 1.

7.6 ERROR MESSAGES

When EDLIN finds an error, one of the following error messages is displayed:

Cannot edit .BAK file—rename file

Cause: You attempted to edit a file with a filename extension of BAK. .BAK files cannot be edited because this extension is reserved for backup copies.

Cure: If you need the .BAK file for editing purposes, you must either RENAME the file with a different extension; or COPY the .BAK file and give it a different filename extension.

No room in directory for file

Cause: When you attempted to create a new file, either the file directory was full or you specified an illegal disk drive or an illegal filename.

Cure: Check the command line that started EDLIN for illegal filename and illegal disk drive entries. If the command is no longer on the screen and if you have not yet typed a new command, the EDLIN start command can be recovered by pressing the <COPYALL> key.

If this command line contains no illegal entries, run the CHKDSK program for the specified disk drive. If the status report shows that the disk directory is full, remove the disk. Insert and format a new disk.

Entry error

Cause: The last command typed contained a syntax error.

Care: Retype the command with the correct syntax and press <ENTER>.

Line too long

Cause: During a Replace command, the string given as the replacement caused the line to expand beyond the limit of 253 characters. EDLIN aborted the Replace command.

Cure: Divide the long line into two lines, then try the Replace command twice.

Disk Full—file write not completed

Cause: You gave the End command, but the disk did not contain enough free space for the whole file. EDLIN aborted the E command and returned you to the operating system. Some of the file may have been written to the disk.

Cure: Only a portion (if any) of the file has been saved. You should probably delete that portion of the file and restart the editing session. The file will not be available after this error. Always be sure that the disk has sufficient free space for the file to be written to disk *before* you begin your editing session.

Incorrect DOS version

Cause: You attempted to run EDLIN under a version of MS-DOS that was not 2.0 or higher.

Cure: You must make sure that the version of MS-DOS that you are using is 2.0 or higher.

Invalid drive name or file

Cause: You have not specified a valid drive or filename when starting EDLIN.

Cure: Specify the correct drive or filename.

File name must be specified

Cause: You did not specify a filename when you started EDLIN.

Cure: Specify a filename.

Invalid Parameter

Cause: You specified a switch other than /B when starting EDLIN.

Cure: Specify the /B switch when you start EDLIN.

Insufficient memory

Cause: There is not enough memory to run EDLIN.

Cure: You must free some memory by writing files to disk or by deleting files before restarting EDLIN.

File not found

Cause: The filename specified during a Transfer command was not found.

Cure: Specify a valid filename when issuing a Transfer command.

Must specify destination number

Cause: A destination line number was not specified for a Copy or Move command.

Cure: Reissue the command with a destination line number.

Not enough room to merge the entire file

Cause: There was not enough room in memory to hold the file during a Transfer command.

Cure: You must free some memory by writing some files to disk or by deleting some files before you can transfer this file.

File creation error

Cause: The EDLIN temporary file cannot be created.

Cure: Check to make sure that the directory has enough space to create the temporary file. Also, make sure that the file does not have the same name as a subdirectory in the directory where the file to be edited is located.

CHAPTER 8

THE LINKER PROGRAM (MS-LINK)

Introduction
Overview Of MS-LINK
Definitions You'll Need To Know
Files That MS-LINK Uses
    Input File Extensions
    Output File Extensions
    VM.TMP (Temporay) File
How To Start MS-LINK
    Method 1: Prompts
    Method 2: Command Line
    Method 3: Response File
Command Characters
Command Prompts
MS-LINK Switches
Sample MS-LINK Session
Error Messages

8.1 INTRODUCTION

In this chapter you will learn about MS-LINK. It is recommended that you read the entire chapter before you use /MS-LINK.

NOTE

If you are not going to compile and link programs, you do not need to read this chapter.

The MS-DOS linker (called MS-LINK) is a program that:

Combines separately produced object modules into one relocatable load module — a program you can run

Searches library files for definitions of unresolved external references

Resolves external cross-references

Produces a listing that shows both the resolution of external references and error messages

8.2 OVERVIEW OF MS-LINK

When you write a program, you write it in source code. This source code is passed through a compiler which produces object modules. The object modules must be passed through the link process to produce machine language that the computer can understand directly. This machine language is in the form required for running programs.

You may wish to link (combine) several programs and run them together. Each of your programs may refer to a symbol that is defined in another object module. This reference is called an *external reference*.

MS-LINK combines several object modules into one relocatable load module, or Runfile (called an .EXE or Executable file). As it combines modules, MS-LINK makes sure that all external references between object modules are defined. MS-LINK can search several library files for definitions of any external references that are not defined in the object modules.

MS-LINK also produces a List file that shows external references resolved, and it also displays any error messages.

MS-LINK uses available memory as much as possible. When available memory is exhausted, MS-LINK creates a temporary disk file named VM.TMP.

Figure 9 illustrates the various parts of the MS-LINK operation.

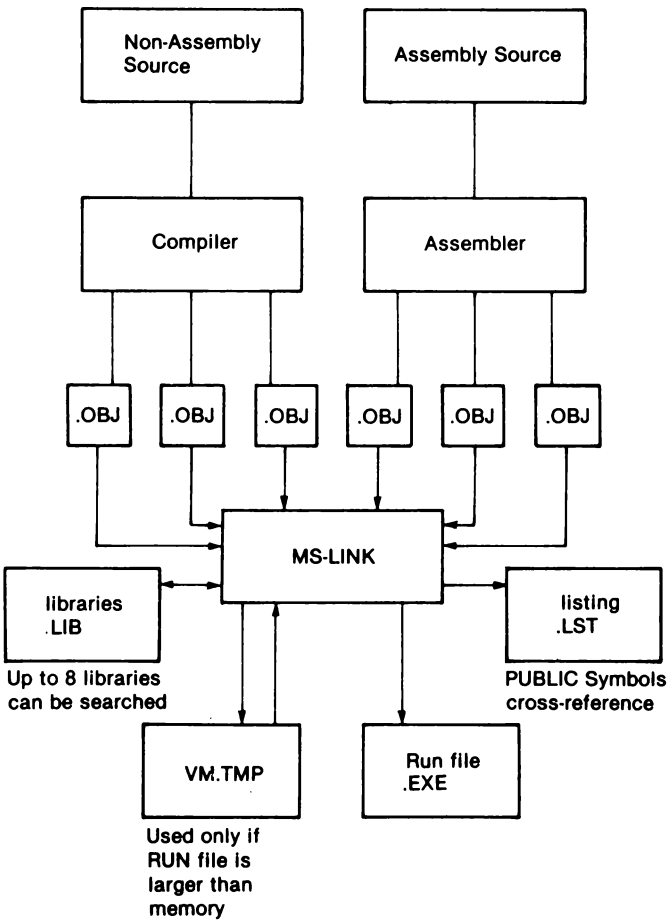


Figure 9. The MS-LINK Operation

8.3 DEFINITIONS YOU'LL NEED TO KNOW

Some of the terms used in this chapter are explained below to help you understand how MS-LINK works. Generally, if you are linking object modules compiled from BASIC, Pascal, or a high-level language, you will not need to know these terms. If you are writing and compiling programs in assembly language, however, you will need to understand MS-LINK and the definitions described below.

In MS-DOS, memory can be divided into segments, classes, and groups. Figure 10 illustrates these concepts.

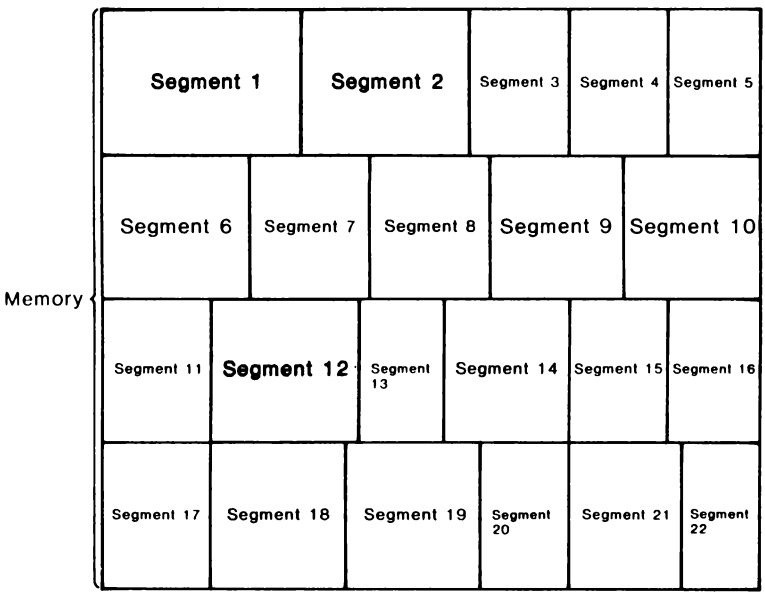


Figure 10. How Memory Is Divided

Example:

	Segment Name	Segment Class Name
Segment 1	PROG.1	CODE
Segment 2	PROG.2	CODE
Segment 12	PROG.3	DATA

Note that segments 1, 2 , and 12 have different segment names but may or may not have the same segment class name. Segments 1, 2 , and 12 form a *group* with a group address of the lowest address of segment 1 (i.e., the lowest address in memory).

Each segment has a segment name and a *class name*. MS-LINK loads all segments into memory by class name from the first segment encountered to the last. All segments assigned to the same class are loaded into memory contiguously.

During processing, MS-LINK references segments by their addresses in memory (where they are located). MS-LINK does this by finding groups of segments.

A *group* is a collection of segments that fit within a 64K byte area of memory. The segments do not need to be contiguous to form a group (see illustration). The address of any group is the lowest address of the segments in that group. At link time, MS-LINK analyzes the groups, then references the segments by the address in memory of that group. A program may consist of one or more groups.

If you are writing in assembly language, you may assign the group and class names in your program. In high-level languages (BASIC, COBOL, FORTRAN, Pascal), the naming is done automatically by the compiler.

8.4 FILES THAT MS-LINK USES

MS-LINK: Works with one or more input files

Produces two output files

May create a temporary disk file

May be directed to search up to eight library files

For each type of file, the user may give a three-part file specification. The format for MS-LINK file specifications is the same as that of a disk file:

[*d:*]<*filename*>[<.*ext*>]

where:

d: is the drive designation. Permissible drive designations for MS-LINK are A: through O:. The colon is always required as part of the drive designation.

filename is any legal filename of one to eight characters.

.ext is a one- to three-character extension to the filename. The period is always required as part of the extension.

8.4.1 Input File Extensions

If no filename extensions are given in the input (object) file specifications, MS-LINK will recognize the following extensions by default:

.OBJ Object
.LIB Library

8.4.2 Output File Extensions

MS-LINK appends the following default extensions to the output (Run and List) files:

.EXE Run (may not be overridden)
.MAP List (may be overridden)

8.4.3 VM.TMP (Temporary) File

MS-LINK uses available memory for the link session. If the files to be linked create an output file that exceeds available memory, MS-LINK will create a temporary file, name it VM.TMP, and put it on the disk in the default drive. If MS-LINK creates VM.TMP, it will display the message:

```
VM.TMP has been created.
Do not change diskette in drive, <d:>
```

Once this message has been displayed, you must not remove the disk from the default drive until the link session ends. If the disk is removed, the operation of MS-LINK will be unpredictable, and MS-LINK might display the error message:

```
Unexpected end of file on VM.TMP
```

The contents of VM.TMP are written to the file named following the Run File: prompt. VM.TMP is a working file only and is deleted at the end of the linking session.

WARNING

Do not use VM.TMP as a file name for any file. If you have a file named VM.TMP on the default drive and MS-LINK requires the VM.TMP file, MS-LINK will delete the VM.TMP already on disk and create a new VM.TMP. Thus, the contents of the previous VM.TMP file will be lost.

8.5 HOW TO START MS-LINK

MS-LINK requires two types of input: a command to start MS-LINK and responses to command prompts. In addition, seven switches control MS-LINK features. Usually, you will type all the commands to MS-LINK on the terminal keyboard. As an option, answers to the command prompts and any switches may be contained in a response file. Command characters can be used to assist you while giving commands to MS-LINK.

MS-LINK may be started in any of three ways. The first method is to type the commands in response to individual prompts. In the second method, you type all commands on the line used to start MS-LINK. To start MS-LINK by the third method, you must create a response file that contains all the necessary commands and tell MS-LINK where that file is when you start MS-LINK.

Summary of Methods to Start MS-LINK

Method 1	LINK
Method 2	LINK *<filenames>*[*/switches*]
Method 3	LINK @ *<filespec>*

8.5.1 Method 1: Prompts

To start MS-LINK with Method 1, type:

```
LINK
```

MS-LINK will be loaded into memory. MS-LINK will then display four text prompts that appear one at a time. You answer the prompts to command MS-LINK to perform specific tasks.

At the end of each line, you may type one or more switches, preceded by the switch character, a forward slash. The command prompts are summarized below and described in more detail in the "Command Prompts" section.

PROMPT	RESPONSES
Object Modules [.OBJ]:	List.OBJ files to be linked. They must be separated by blank spaces or plus signs (+). If a plus sign is the last character typed, the prompt will reappear. There is no default; a response is required.
Run File [Object-file.EXE]:	Give filename for executable object code. The default is first-object-filename.EXE. (You cannot change the output extension.)
List File [NUL.MAP]:	Give filename for listing. The default is NUL file.
Libraries [.LIB]:	List filenames to be searched, separated by blank spaces or plus signs (+). If a plus sign is the last character typed, the prompt will reappear. The default is to search for default/libraries in the object modules. (Extensions will be changed to .LIB.)

8.5.2 Method 2: Command Line

To start MS-LINK using Method 2, type all commands on one line. The entries following LINK are responses to the command prompts. The entry fields for the different prompts must be separated by commas. Use the following syntax:

LINK <*object-list*>, <*runfile*>, <*listfile*>, <*lib-list*>
[*lswitch*...]

The entries following LINK are responses to the command prompts. The entry fields for the different prompts must be separated by commas.

where: *object-list* is a list of object modules, separated by plus signs.

runfile is the name of the file to receive the executable output.

listfile is the name of the file to receive the listing.

lib-list is a list of library modules to be searched.

/switch refers to optional switches, which may be placed following any of the response entries (just before any of the commas or after the <*lib-list*>, as shown).

To select the default for a field, simply type a second comma with no spaces between the two commas.

Example: LINK FUN+TEXT+TABLE+CARE/P/M,,
FUNLIST,COBLIB.LIB

This command causes MS-LINK to be loaded, then the object modules FUN.OBJ, TEXT.OBJ, TABLE.OBJ, and CARE.OBJ are loaded. MS-LINK then pauses (as a result of using the /P switch). MS-LINK links the object modules when you press any key, and produces a global symbol map (the /M switch); defaults to FUN.EXE Run file; creates a List file named FUNLIST.MAP; and searches the Library file COBLIB.LIB.

8.5.3 Method 3: Response File

To start MS-LINK with Method 3, type:

LINK @ <*filespec*>

where: *filespec* is the name of a response file. A response file contains answers to the MS-LINK prompts (shown in Method 1) and may also contain any of the switches. When naming a response file, the use of filename extensions is optional. Method 3 permits the command that starts MS-LINK to be entered from the keyboard or within a batch file without requiring you to take any further action.

To use this option, you must create a response file containing several lines of text, each of which is the response to an MS-LINK prompt. The responses must be in the same order as the MS-LINK prompts discussed in Method 1. If desired, a long response to the Object Modules: or Libraries: prompt may be typed on several lines by using a plus sign (+) to continue the same response onto the next line.

Use switches and command characters in the response file the same way as they are used for responses typed on the terminal keyboard.

When the MS-LINK session begins, each prompt will be displayed in order with the responses from the response file. If the response file does not contain answers for all the prompts, (in the form of filenames, the semicolon command character or carriage returns), MS-LINK will display the prompt which does not have a response, then wait for you to type a legal response. When a legal response has been typed, MS-LINK continues the link session.

Example:

```
FUN TEXT TABLE CARE
/PAUSE/MAP
FUNLIST
COBLIB.LIB
```

This response file tells MS-LINK to load the four object modules named FUN, TEXT, TABLE, and CARE. MS-LINK pauses before producing a public symbol map to permit you to swap disks (see discussion under /PAUSE in the "Switches" section before using this feature). When you press any key, the output files will be named FUN.EXE and FUNLIST.MAP. MS-LINK will search the library file COBLIB.LIB, and will use the default settings for the switches.

8.6 COMMAND CHARACTERS

MS-LINK provides three command characters.

Plus sign Use the plus sign (+) to separate entries and to extend the current line in response to the Object Modules: and Libraries: prompts. (A blank space may be used to separate object modules.) To type a large number of responses (each may be very long), type a plus sign/<ENTER> at the end of the line to extend it. If the plus sign/<ENTER> is the last entry following these two prompts, MS-LINK will prompt you for more module names. When the Object Modules: or Libraries: prompt appears again, continue to type responses. When all the modules to be linked and libraries to be searched have been listed, be sure the response line ends with a module name and a <ENTER> and not a *plus* sign/<ENTER>.

Example:

```
Object Modules [.OBJ]: FUN TEXT
TABLE CARE + <ENTER>
Object Modules [.OBJ]:
FOO + FLIPFLOP + JUNQUE + <ENTER>
Object Modules [.OBJ]:
CORSAIR<ENTER>
```

Semicolon To select default responses to the remaining prompts, use a single semecolon (;) followed immediately by a carriage return at any time after the first prompt (Run File:). This feature saves time and overrides the need to press a series of <ENTER> keys.

NOTE

Once the semicolon has been typed and entered by pressing the <ENTER> key, you can no longer respond to any of the prompts for that link session. Therefore, do not use the semicolon to skip some prompts.
To skip prompts, use the <ENTER> key.

Example: Object Modules [.OBJ]: FUN TEXT
TABLECARE<ENTER>
Run File [FUN.EXE]: ; <ENTER>

No other prompts will appear, and MS-LINK will use the default values (including FUN.MAP for the List file).

<CONTROL-C> Use the <CONTROL-C> key to abort the link session at any time. If you type an erroneous response, such as the wrong filename or an incorrectly spelled filename, you must press <CONTROL-C> to exit MS-LINK then restart MS-LINK. If the error has been typed but you have not pressed the <ENTER> key, you may delete the erroneous characters with the backspace key, but for that line only.

8.7 COMMAND PROMPTS

MS-LINK asks you for responses to four text prompts. When you have typed a response to a prompt and pressed <ENTER>, the next prompt appears. When the last prompt has been answered, MS-LINK begins linking automatically without further command. When the link session is finished, MS-LINK exits to the operating system. When the operating system prompt appears, MS-LINK has finished successfully. If the link session is unsuccessful, MS-LINK will display the appropriate error message.

MS-LINK prompts you for the names of Object, Run and List files, and for Libraries. The prompts are listed in order of appearance. The default response is shown in square brackets ([ ]) following the prompt, for prompts which can default to preset responses. The Object Modules: prompt has no preset filename response and requires you to type a filename.

Object Modules [.OBJ]:

Type a list of the object modules to be linked. MS-LINK assumes by default that the filename extension is .OBJ. If an object module has any other filename extension, the extension must be given. Otherwise, the extension may be omitted.

Modules must be separated by plus signs(+).

Remember that MS-LINK loads segments into classes in the order encountered. You can use this information to set the order in which the object modules will be read by MS-LINK.

Run File [First-Object-filename.EXE]:

Typing a filename will create a file for storing the Run (executable) file that results from the link session. All Run files receive the filename extension .EXE, even if you specify an extension other than .EXE.

If no response is typed to the Run File: prompt, MS-LINK uses the first filename typed in response to the Object Modules: prompt as the RUN filename.

Example: Run File [FUN.EXE]: B:PAYROLL/P

This response directs MS-LINK to create the Run file PAYROLL.EXE on drive B:. Also, MS-LINK will pause, which allows you to insert a new disk to receive the Run file.

List File [NUL.MAP]:

The List file contains an entry for each segment in the input (object) modules. Each entry also shows the addressing in the Run file.

The default response is the Run filename with the default NUL.MAP.

Libraries [.LIB]:

The valid responses are up to eight library filenames or simply a carriage return. (A carriage return means default library search.) Library files must have been created by a library utility. MS-LINK assumes by default that the filename extension is .LIB for library files.

Library filenames must be separated by blank spaces or plus signs (+).

MS-LINK searches library files in the order listed to resolve external references. When it finds the module that defines the external symbol, MS-LINK processes that module as another object module.

If MS-LINK cannot find a library file on the disks in the disk drives, it will display the message:

Cannot find library <*library-name*>
Enter new file spec:

Press the letter for the drive designation (for example, B).

8.8 MS-LINK SWITCHES

The seven MS-LINK switches control various MS-LINK functions. Switches must be typed at the end of a prompt response, regardless of which method is used to start MS-LINK. Switches may be grouped at the end of any response, or may be scattered at the end of several. If more than one switch is typed at the end of one response, each switch must be preceded by a forward slash (/).

All switches may be abbreviated. The only restriction is that an abbreviation must be sequential from the first letter through the last typed; no gaps or transpositions are allowed. For example:

Legal	*Illegal*
/D	/DSL
/DS	/DAL
/DSA	/DLC
/DSALLOCA	/DSALLOCT

/DSALLOCATE

Using the /DSALLOCATE switch tells MS-LINK to load all data at the high end of the Data Segment. Otherwise, MS-LINK loads all data at the low end of the Data Segment. At runtime, the DS pointer is set to the lowest possible address to allow the entire DS segment to be used. Use of the /DSALLOCATE switch in combination with the default load low (that is, the /HIGH switch is *not* used) permits the user application to dynamically allocate any available memory below the area specifically allocated within D Group, yet to remain addressable by the same DS pointer. This dynamic allocation is needed for Pascal and FORTRAN programs.

NOTE

Your application program may dynamically allocate up to 64K bytes (or the actual amount of memory available) less the amount allocated within D Group.

/HIGH Use of the /HIGH switch causes MS-LINK to place the Run file as high as possible in memory. Otherwise, MS-LINK places the Run file as low as possible.

IMPORTANT

Do not use the /HIGH switch with Pascal or FORTRAN programs.

/LINENUMBERS

The /LINENUMBERS switch tells MS-LINK to include in the List file the line numbers and addresses of the source statements in the input modules. Otherwise, line numbers are not included in the List file.

NOTE

Not all compilers produce object modules that contain line number information. In these cases, of course, MS-LINK cannot include line numbers.

/MAP /MAP directs MS-LINK to list all public (global) symbols defined in the input modules. If /MAP is not given, MS-LINK will list only errors (including undefined globals).

The symbols are listed alphabetically. For each symbol, MS-LINK lists its value and its segment: offset location in the Run file. The symbols are listed at the end of the List file.

/PAUSE The /PAUSE switch causes MS-LINK to pause in the link session when the switch is encountered. Normally, MS-LINK performs the linking session from beginning to end without stopping. This switch allows the user to swap the disks before MS-LINK outputs the Run (.EXE) file.

When MS-LINK encounters the /PAUSE switch, it displays the message:

```
About to generate .EXE file
Change disks <hit any key>
```

MS-LINK resumes processing when the user presses any key.

CAUTION

Do not remove the disk which will receive the List file, or the disk used for the VM.TMP file, if one has been created.

/STACK:<number>

number represents any positive numeric value (in hexadecimal radix) up to 65536 bytes. If a value from 1 to 511 is typed, MS-LINK will use 512. If the /STACK switch is not used for a link session, MS-LINK will calculate the necessary stack size automatically.

All compilers and assemblers should provide information in the object modules that allow the linker to compute the required stack size.

At least one object (input) module must contain a stack allocation statement. If not, MS-LINK will display the following error message:

Warning: No STACK segment

/NO /NO is short for NODEFAULTLIBRARYSEARCH. This switch tells MS-LINK *not* to search the default (product) libraries in the object modules. For example, if you are linking object modules in Pascal, specifying the /NO switch tells MS-LINK not to automatically search the library named PASCAL.LIB to resolve external references.

8.9 SAMPLE MS-LINK SESSION

This sample shows you the type of information that is displayed during an MS-LINK session.

In response to the MS-DOS prompt, type:

```
LINK
```

The system displays the following messages and prompts (your answers are underlined):

```
        Microsoft Object Linker V.2.1(830822)
(C)  Copyright 1983 by Microsoft Inc.

Object Modules [.OBJ]: IO SYSINIT
Run File [IO.EXE]:
List File [NUL.MAP]: IO/MAP
Libraries [.LIB]:_
```

NOTES

1. By specifying /MAP, you get both an alphabetic listing and a chronological listing of public symbols.

2. By responding PRN to the List File: prompt, you can redirect your output to the printer.

3. By specifying the /LINE switch, MS-LINK gives you a listing of all line numbers for all modules. (Note that the /LINE switch can generate a large volume of output.)

4. By pressing <ENTER> in response to the Libraries: prompt, an automatic library search is performed.

Once MS-LINK locates all libraries, the linker map displays a list of segments in the order of their appearance within the load module. The list might look like this:

```
Start     Stop      Length    Name
00000H    009ECH    09EDH     CODE
009F0H    01166H    0777H     SYSINITSEG
```

The information in the Start and Stop columns shows the 20-bit hex address of each segment relative to location zero. Location zero is the beginning of the load module.

The addresses displayed are not the absolute addresses where these segments are loaded.

Because the /MAP switch was used, MS-LINK displays the public symbols by name and value. For example:

```
ADDRESS         PUBLICS BY NAME
009F:0012       BUFFERS
009F:0005       CURRENT DOS LOCATION
009F:0011       DEFAULT DRIVE
009F:000B       DEVICE LIST
009F:0013       FILES
009F:0009       FINAL DOS LOCATION
009F:000F       MEMORY SIZE
009F:0000       SYSINIT

ADDRESS         PUBLICS BY VALUE
009F:0000       SYSINIT
009F:0005       CURRENT DOS LOCATION
009F:0009       FINAL DOS LOCATION
009F:000B       DEVICE LIST
009F:000F       MEMORY SIZE
009F:0011       DEFAULT DRIVE
009F:0012       BUFFERS
009F:0013       FILES
```

8.10 ERROR MESSAGES

All errors cause the link session to abort. After cause has been found and corrected, MS-LINK must be rerun. The following error messages are displayed by MS-LINK:

Attempt to access data outside of segment bounds
Input File: <filename> pos: xxx Record type: xxx
There is probably a bad Object file.

Bad numeric parameter
Numeric value is not in digits.

Cannot open temporary file
MS-LINK is unable to create the file VM.TMP because the disk directory is full. Insert a new disk. Do not remove the disk that will receive the List.MAP file.

Error: DUP record too complex
DUP record in assembly language module is too complex. Simplify DUP record in assembly language program.

Error: fixup offset exceeds field width
An assembly language instruction refers to an address with a short instruction instead of a long instruction. Edit assembly language source and reassemble.

Input file read error
There is probably a bad Object file.

Invalid object module
An object module(s) is incorrectly formed or incomplete (as when assembly is stopped in the middle).

Symbol defined more than once
MS-LINK found two or more modules that define a single symbol name.

Program size or number of segments exceeds capacity of linker
The total size may not exceed 384K bytes and the number of segments may not exceed 255.

Requested stack size exceeds 64K

Specify a size greater than or equal to 64K bytes with the /STACK switch.

Segment size exceeds 64K

64K bytes is the addressing system limit.

Symbol table capacity exceeded

Very many and/or very long names were typed, exceeding the limit of approximately 25K bytes.

Too many external symbols in one module

The limit is 256 external symbols per module.

Too many groups

The limit is 10 groups.

Too many libraries specified

The limit is 8 libraries.

Too many PUBLIC symbols

The limit is 1024 public symbols.

Too many segments or classes

The limit is 256 (segments and classes taken together).

Unresolved externals: <list>

The external symbols listed have no defining module among the modules or library files specified.

VM read error

This is a disk error; it is not caused by MS-LINK.

Warning: no stack segment

None of the object modules specified contains a statement allocating stack space, but you typed the /STACK switch.

Warning: segment of absolute or unknown type

There is a bad object module or an attempt has been made to link modules that MS-LINK cannot handle (e.g., an absolute object module).

Write error in TMP file

No more disk space remains to expand the VM.TMP file.

Write error on run file

Usually, there is not enough disk space for the Run file.

CHAPTER 9

THE DEBUGGER PROGRAM(DEBUG)

Introduction
How To Start DEBUG
    Method 1: DEBUG
    Method 2: Command Line
Command Information
Parameters
Error Messages

9.1 INTRODUCTION

The Microsoft DEBUG Utility (DEBUG) is a debugging program that provides a controlled testing environment for binary and executable object files. Note that EDLIN is used to alter source files; DEBUG is EDLIN's counterpart for binary files. DEBUG eliminates the need to reassemble a program to see if a problem has been fixed by a minor change. It allows you to alter the contents of a file or the contents of a CPU register, and then to immediately reexecute a program to check on the validity of the changes.

All DEBUG commands may be aborted at any time by pressing <CONTROL-C>. <CONTROL-S> suspends the display, so that you can read it before the output scrolls away. Entering any key other than <CONTROL-C> or <CONTROL-S> restarts the display. All of these commands are consistent with the control character functions available at the MS-DOS command level.

9.2 HOW TO START DEBUG

DEBUG may be started two ways. By the first method, you type all commands in response to the DEBUG prompt (a hyphen).By the second method, you type all commands on the line used to start DEBUG.

Summary of Methods to Start DEBUG

Method 1	DEBUG
Method 2	DEBUG [<*filespec*> [<*arglist*>]]

9.2.1 Method 1: DEBUG

To start DEBUG using method 1, type:

```
DEBUG
```

DEBUG responds with the hyphen (-) prompt, signaling that it is ready to accept your commands. Since no filename has been specified, current memory, disk sectors, or disk files can be worked on by using other commands.

Warnings

1. When DEBUG (Version 2.0) is started, it sets up a program header at offset 0 in the program work area. On previous versions of DEBUG, you could overwrite this header. You can still overwrite the default header if no <*filespec*> is given to DEBUG. If you are debugging a .COM or .EXE file, however, do not tamper with the program header below address 5CH, or DEBUG will terminate.

2. Do not restart a program after the "Program terminated normally" message is displayed. You must reload the program with the N and L commands for it to run properly.

9.2.2 Method 2: Command Line

To start DEBUG using a command line, type:

DEBUG [<*filespec*> [<*arglist*>]]

For example, if a <*filespec*> is specified, then the following is a typical command to start DEBUG:

DEBUG FILE.EXE

DEBUG then loads FILE.EXE into memory starting at 100 hexadecimal in the lowest available segment. The BX:CX registers are loaded with the number of bytes placed into memory.

An <*arglist*> may be specified if <*filespec*> is present. The <*arglist*> is a list of filename parameters and switches that are to be passed to the program <*filespec*>. Thus, when <*filespec*> is loaded into memory, it is loaded as if it had been started with the command:

<*filespec*> <*arglist*>

Here, <*filespec*> is the file to be debugged, and the <*arglist*> is the rest of the command line that is used when <*filespec*> is invoked and loaded into memory.

9.3 COMMAND INFORMATION

Each DEBUG command consists of a single letter followed by one or more parameters. Additionally, the control characters and the special editing functions described in this book.

If a syntax error occurs in a DEBUG command, DEBUG reprints the command line and indicates the error with an up-arrow (^) and the word "Error."

For example:

```
dcs:100 cs:110
        ^ Error
```

Any combination of uppercase and lowercase letters may be used in commands and parameters.

The DEBUG commands are summarized in Table 10.1 and are described in detail, with examples, following the description of command parameters.

Table 9.1 DEBUG Commands

DEBUG Command	Function
A [<*address*>]	Assemble
C <*range*> <*address*>	Compare
D [<*range*>]	Dump
E <*address*> [<*list*>]	Enter
F <*range*> <*list*>	Fill
G [= <*address*> [<*address*>...]]	Go
H <*value*> <*value*>	Hex
I <*value*>	Input
L [<*address*> [<*drive*> <*record*> <*record*>]]	Load
M <*range*> <*address*>	Move
N <*filename*> [<*filename*>]	Name
O <*value*> <*byte*>	Output
Q	Quit
R [<*register-name*>]	Register
S <*range*> <*list*>	Search
T [= <*address*>] [<*value*>]	Trace
U [<*range*>]	Unassemble
W [<*address*> [<*drive*> <*record*> <*record*>]]	Write

9.4 PARAMETERS

All DEBUG commands accept parameters, except the Quit command. Parameters may be separated by delimiters (spaces or commas), but a delimiter is required only between two consecutive hexadecimal values. Thus, the following commands are equivalent:

```
dcs:100 110
d cs:100 110
d,cs:100,110
```

PARAMETER	DEFINITION
<drive>	A one-digit hexadecimal value to indicate which drive a file will be loaded from or written to. The valid values are 0-1. These values designate the drives as follows: 0 = A, 1 = B.
<byte>	A two-digit hexadecimal value to be placed in or read from an address or register.
<record>	A 1- to 3-digit hexadecimal value used to indicate the logical record number on the disk and the number of disk sectors to be written or loaded. Logical records correspond to sectors. However, their numbering differs since they represent the entire disk space.
<value>	A hexadecimal value up to four digits used to specify a port number or the number of times a command should repeat its functions.
<address>	A two-part designation consisting of either an alphabetic segment register designation or a four-digit segment address plus an offset value. The segment designation or segment address may be omitted, in which case the default segment is used. DS is the default segment for all commands except G, L, T, U, and W, for which the default segment is CS. All numeric values are hexadecimal.

For example:

```
DCS:0100
D04BA:0100
```

The colon is required between a segment designation (whether numeric or alphabetic) and an offset.

<range>

Two *<address>*es: e.g., *<address>* *<address>*; or one *<address>*, an L, and a *<value>*: e.g., *<address>* L *<value>* where *<value>* is the number of lines the command should operate on, and L80 is assumed. The last form cannot be used if another hex value follows the *<range>*, since the hex value would be interpreted as the second *<address>* of the *<range>*.

Examples:

```
DCS:100 110
DCS:100 L 10
DCS:100
```

The following is illegal:

```
CS:100 CS:110
       ^ Error
```

The limit for *<range>* is 10000 hex. To specify a *<value>* of 10000 hex within four digits, type 0000 (or 0).

<list>

A series of *<byte>* values or of *<string>*s. *<list>* must be the last parameter on the command line.

Example:

```
fcs:  100  L5  42 45 52 54 41
```

<string>

Any number of characters enclosed in quote marks. Quote marks may be either single (') or double ("). If the delimiter quote marks must appear within a *<string>*, the quote marks must be doubled. For example, the following strings are legal:

```
'This is a "string" is okay.'
'This is a "string" is okay.'
```

However, this string is illegal:

```
'This is a 'string' is not.'
```

Similarly, these strings are legal:

```
"This is a 'string' is okay."
"This is a ""string"" is okay."
```

However, this string is illegal:

```
"This is a "string " is not."
```

Note that the double quote marks are not necessary in the following strings:

```
"This is a "string" is not necessary."
'This is a ""string "" is not necessary.'
```

The ASCII values of the characters in the string are used as a *<list >* of byte values.

ASSEMBLE

Name: Assemble

Purpose: Assembles 8086/8087/8088 mnemonics directly into memory.

Syntax: A [<*address*>]

Comments: If a syntax error is found, DEBUG responds with

^Error

and redisplays the current assembly address.

All numeric values are hexadecimal and must be entered as 1-4 characters. Prefix mnemonics must be specified in front of the opcode to which they refer. They may also be entered on a separate line.

The segment override mnemonics are CS:, DS:, ES:, and SS:. The mnemonic for the far return is RETF. String manipulation mnemonics must explicity state the string size. For example, use MOVSW to move word strings and MOVSB to move byte strings.

The assembler will automatically assemble short, near or far jumps and calls, depending on byte displacement to the destination address. These may be overridden with the NEAR or FAR prefix. For example:

```
0100:0500 JMP   502
                ; a 2-byte short jump
0100:0502 JMP   NEAR 505
                ; a 3-byte near jump
0100:505   JMP FAR 50A
                ;a 5-byte far jump
```

The NEAR prefix may be abbreviated to NE, but the FAR prefix cannot be addreviated.

DEBUG cannot tell whether some operands refer to a word memory location or to a byte memory location. In this case, the data type must be explicitly stated with the prefix "WORD PTR" or "BYTE PTR". Acceptable abbreviations are "WO" and "BY". For example:

```
NEG   BYTE PTR [128]
DEC   WO [SI]
```

DEBUG also cannot tell whether an operand refers to a memory location or to an immediate operand. DEBUG uses the common convention that operands enclosed in square brackets refer to memory. For example:

```
MOV   AX,21      ; Load AX with 21H
MOV   AX,[21]    ; Load AX with the
                 ; contents
                 ; of memory location 21H
```

Two popular pseudo-instructions are available with Assemble. The DB opcode will assemble byte values directly into memory. The DW opcode will assemble word values directly into memory. For example:

```
DB   1,2,3,4, "THIS IS AN EXAMPLE"
DB   'THIS IS A QUOTE: '"
DB   "THIS IS A QUOTE: '"

DW 1000,2000,3000,"BACH"
```

Assemble supports all forms of register indirect commands. For example:

```
ADD    BX,34[BP + 2] . [SI-1]
POP    [BP + DI]
PUSH   [SI]
```

All opcode synonyms are also supported. For example:

```
LOOPZ   100
LOOPE   100

JA      200
JNBE    200
```

For 8087 opcodes, the WAIT or FWAIT must be explicity specified. For example:

```
FWAIT FADD ST,ST(3)
            ; This line will assemble
            ; an FWAIT prefix
LD TBYTE PTR [BX]
            ; This line will not
```

COMPARE

Name: Compare

Purpose: Compares the portion of memory specified by *<range>* to a portion of the same size beginning at *<address>*.

Syntax: C*<range>* *<address>*

Comments: If the two areas of memory are identical, there is no display and DEBUG returns with the MS-DOS prompt. If there are differences, they are displayed in this format:

<address1> *<byte1>* *<byte2>* *<address2>*

Example: The following commands have the same effect:

```
C100, 1FF 300
      or
C100L100 300
```

Each command compares the block of memory from 100 to 1FFH with the block of memory from 300 to 3FFH.

DUMP

Name: Dump

Purpose: Displays the contents of the specified region of memory.

Syntax: D [<*range*>]

Comments: If a range of addresses is specified, the contents of the range are displayed. If the D command is typed without parameters, 128 bytes are displayed at the first address (DS:100) after the address displayed by the previous Dump command.

The dump is displayed in two portions: a hexadecimal dump (each byte is shown in hexadecimal value) and an ASCII dump (the bytes are shown in ASCII characters). Nonprinting characters are denoted by a period (.) in the ASCII portion of the display. Each display line shows 16 bytes with a hyphen between the eighth and ninth bytes. At times, displays are split in this manual to fit them on the page. Each displayed line begins on a 16-byte boundary.

If you type the command:

```
dec:100 10E
```

DEBUG displays the dump in the following format:

```
00EC:0100 42 45 52 54 41 .....4E 44
                          BERTF T.BORLAND
```

If you type the following command:

```
D
```

the display is formatted as described above. Each line of the display begins with an address, incremented by 16 from the address on the previous line. Each subsequent D (typed without parameters) displays the bytes immediately following those last displayed.

If you type the command:

```
DDS:100 L 20
```

the display is formatted as described above, but 20H bytes are displayed.

If then you type the command:

```
DCS:100 115
```

the display is formatted as described above, but all the bytes in the range of lines from 100H to 115H in the CS segment are displayed.

ENTER

Name: Enter

Purpose: Enters byte values into memory at the specified <*address*>.

Syntax: E<*address*> [<*list*>]

Comments: If the optional <*list*> of values is typed, the replacement of byte values occurs automatically. (If an error occurs, no byte values are changed.)

If the <*address*> is typed without the optional <*list*>, DEBUG displays the address and its contents, then repeats the address on the next line and waits for your input. At this point, the Enter command waits for you to perform one of the following actions:

1. Replace a byte value with a value you type. Simply type the value after the current value. If the value typed in is not a legal hexadecimal value or if more than two digits are typed, the illegal or extra character is not echoed.

2. Press the <SPACE> bar to advance to the next byte. To change the value, simply type the new value as described in (1). above. If you space beyond an 8-byte boundary, DEBUG starts a new display line with the address displayed at the beginning.

3. Type a hyphen (-) to return to the preceding byte. If you decide to change a byte behind the current position, typing the hyphen returns the current position to the previous byte. When the hyphen is typed, a new line is started with the address and its byte value displayed.

4. Press <ENTER> to terminate the Enter command. <ENTER> may be pressed to any byte position.

Example: Assume that the following command is typed:

```
ECS: 100
```

DEBUG displays:

```
04BA:0100  EB.__
```

To change this value to 41, type 41 as shown:

```
04BA:0100  EB.41__
```

To step through the subsequent bytes, press the <SPACE> bar to see:

```
04BA:0100  EB.41  10.  00.  BC.__
```

To change BC to 42:

```
04BA:0100  EB.41  10.  00.  BC.42__
```

Now, realizing that 10 should be 6F, type the hyphen as many times as needed to return to byte 0101 (value 10), then replace 10 with 6F:

```
04BA:0100  EB.41  10.  00.  BC.42__
04BA:0102  00.-__
04BA:0101  10.6F__
```

Pressing <ENTER> ends the Enter command and returns to DEBUG command level.

FILL

Name: Fill

Purpose: Fills the addresses in the <*range*> with the values in the <*list*>.

Syntax: F<*range*><*list*>

Comments: If the <*range*> contains more bytes than the number of values in the <*list*>, the <*list*> will be used repeatedly until all bytes in the <*range*> are filled. If the <*list*> contains more values than the number of bytes in the <*range*>, the extra values in the <*list*> will be ignored. If any of the memory in the <*range*> is not valid (bad or nonexistent), the error will occur in all succeeding locations.

Example: Assume that the following command is typed:

```
F04BA:100 L 100 42 45 52 54 41
```

DEBUG fills memory locations 04BA:100 through 04BA:1FF with the bytes specified. The five values are repeated until all 100H bytes are filled.

GO

Name: Go

Purpose: Executes the program currently in memory.

Syntax: G [= <*address*> [<*address*>...]]

Comments: If only the Go command is typed, the program executes as if the program had run outside DEBUG.

If = <*address*> is set, execution begins at the address specified. The equal sign (=) is required, so that DEBUG can distinguish the start = <*address*> from the breakpoint <*address*> es.

With the other optional addresses set, execution stops at the first <*address*> encountered, regardless of that address' position in the list of addresses to halt execution or program branching. When program execution reaches a breakpoint, the registers, flags, and decoded instruction are displayed for the last instruction executed. (The result is the same as if you had typed the Register command for the breakpoint address.)

Up to ten breakpoints may be set. Breakpoints may be set only at addresses containing the first byte of an 8086 opcode. If more than ten breakpoints are set, DEBUG returns the BP Error message.

The user stack pointer must be valid and have 6 bytes available for this command. The G command uses an IRET instruction to cause a jump to the program under test. The user stack pointer is set, and the user flags, Code Segment register, and Instruction Pointer are pushed on the user stack. (Thus, if the user stack is not valid or is too small, the operating system may crash.) An interrupt code (0CCH) is placed at the specified breakpoint address(es).

When an instruction with the breakpoint code is encountered, all breakpoint addresses are restored to their original instructions. If execution is not halted at one of the breakpoints, the interrupt codes are not replaced with the original instructions.

Example: Assume that the following command is typed:

```
GCS:7550
```

The program currently in memory executes up to the address 7550 in the CS segment. DEBUG then displays registers and flags, after which the Go command is terminated.

After a breakpoint has been encountered, if you type the Go command again, then the program executes just as if you had typed the filename at the MS-DOS command level. The only difference is that program execution begins at the instruction after the breakpoint rather than at the usual start address.

HEX

Name: Hex

Purpose: Performs hexadecimal arithmetic on the two parameters specified.

Syntax: H <*value*> <*value*>

Comments: First, DEBUG adds the two parameters, then subtracts the second parameter from the first. The results of the arithmetic are displayed on one line; first the sum, then the difference.

Example: Assume that the following command is typed:

```
H19F 10A
```

DEBUG performs the calculations and then displays the result:

```
02A9  0095
```

INPUT

Name: Input

Purpose: Inputs and displays one byte from the port specified by *<value>*.

Syntax: I *<value>*

Comments: A 16-bit port address is allowed.

Example: Assume that you type the following command:

```
I2F8
```

Assume also that the byte at the port is FFH. DEBUG inputs the byte and displays the value:

```
FF
```

LOAD

Name: Load

Purpose: Loads a file into memory.

Syntax: L [<*address*>[<*drive*> <*record*> <*record*>]]

Comments: Set BX:CX to the number of bytes read. The file must have been named either when DEBUG was started or with the N command. Both the DEBUG invocation and the N command format a filename properly in the normal format of a file control block at CS:5C.

If the L command is typed without any parameters, DEBUG loads the file into memory beginning at address CS:100 and sets BX:CX to the number of bytes loaded. If the L command is typed with an address parameter, loading begins at the memory <*address*> specified. If L is typed with all parameters, absolute disk sectors are loaded, not a file. The <*record*>s are taken from the <*drive*> specified (the drive designation is numeric here—0 = A:, 1 = B:. DEBUG begins loading with the first <*record*> specified, and continues until the number of sectors specified in the second <*record*> has been loaded.

Example: Assume that the following commands are typed:

```
A>DEBUG  NFILE.COM
```

Now, to load FILE.COM, type:

```
L
```

DEBUG loads the file and then dispalys the DEBUG prompt. Assume that you want to load only portions of a file of certain records from a disk. To do this, type:

```
L04BA:100 2 0F 6D
```

DEBUG then loads 109 (6D hex) records beginning with logical record number 15 into memory beginning at address 04BA:0100. When the records have been loaded, DEBUG simply returns the - prompt.

If the file has a .EXE extension, it is relocated to the load address specified in the header of the .EXE file: the <*address*> parameter is always ignored for .EXE files. The header itself is stripped off the .EXE file before it is loaded into memory. Thus the size of an .EXE file on disk will differ from its size in memory.

If the file named by the Name commmand or specified when DEBUG is started is a .HEX file, then typing the L command with no parameters causes DEBUG to load the file beginning at the address specified in the .HEX file. If the L command includes the option <*address*>, DEBUG adds the <*address*> specified in the L command to the address found in the .HEX file to determine the start address for loading the file.

MOVE

Name: Move

Purpose: Moves the block of memory specified by *<range>* to the location beginning at the *<address>* specified.

Syntax: M *<range>* *<address>*

Comments: Overlapping moves (i.e.,moves where part of the block overlaps some of the current addresses) are always performed without loss of data. Addresses that could be overwritten are moved first. The sequence for moves from higher addresses to lower addresses is to move the data beginning at the block's lowest address and then to work towards the highest. The sequence for moves from lower addresses to higher addresses is to move the data beginning at the block's highest address and to work towards the lowest.

Note that if the addresses in the block being moved will not have new data written to them, the data there before the move will remain. The M command copies the data from one area into another, in the sequence described, and writes over the new addresses. This is why the sequence of the move is important.

Example: Assume that you type:

```
MCS:100 110 CS:500
```

DEBUG first moves address CS:110 to address CS:510, then CS:10F to CS:50F, and so on until CS:100 is moved to VS:500. You should type the D command, using the *<address>* typed for the M command, to review the results of the move.

NAME

Name: Name

Purpose: Sets filenames.

Syntax: N <*filename*> [<*filename*>...]

Comments: The Name command performs two functions. First, Name is used to assign a filename for a later Load or Write command. Thus, if you start DEBUG without naming any file to be debugged, then the N <*filename*> command must be typed before a file can be loaded. Second, Name is used to assign filename parameters to the file being debugged. In this case, Name accepts a list of parameters that are used by the file being debugged.

These two functions overlap. Consider the following set of DEBUG commands:

```
-NFILE1.EXE
-L
-G
```

Because of the effects of the Name command, Name will perform the following steps:

1. (N)ame assigns the filename FILE1.EXE to the filename to be used in any later Load or Write commands.

2. (N)ame also assigns the filename FILE1.EXE to the first filename parameter used by any program that is later debugged.

3. (L)oad loads FILE1.EXE into memory.

4. (G)o causes FILE1.EXE to be executed with FILE1.EXE as the single filename parameter (that is, FILE1.EXE is executed as if FILE1.EXE had been typed at the command level).

A more useful chain of commands might look like this:

```
-NFILE1.EXE
-L
-NFILE2.DAT FILE3.DAT
-G
```

Here, Name sets FILE1.EXE as the filename for the subsequent load command. The Load command loads FILE1.EXE into memory, and then the Name command is used again, this time to specify the parameters to be used by FILE1.EXE. Finally, when the Go command is executed, FILE1.EXE is executed as if FILE1 FILE2.DAT FILE3.DAT had been typed at the MS-DOS command level. Note that if a Write command were executed at this point, then FILE1.EXE — the file being debugged — would be saved with the name FILE2.DAT! To avoid such undesired results, you should always execute a Name command before either a Load or a Write.

There are four regions of memory that can be affected by the Name command:

CS:5C	FCB for file 1
CS:6C	FCB for file 2
CS:80	Count of characters
CS:81	All characters typed

A File Control Block (FCB) for the first filename parameter given to the Name command is set up at CS:5C. If a second filename parameter is typed, then an FCB is set up for it beginning at CS:6C. The number of characters typed in the Name command (exclusive of the first character, "N") is given at location CS:80. The actual stream of characters given by the Name command (again, exclusive of the letter "N") begins at CS:81. Note that this stream of characters may contain switches and delimiters that would be legal in any command typed at the MS-DOS command level.

Example: A typical use of the Name command is:

```
A>DEBUG PROG.COM
-NPARAM1 PARAM2/C
-G
-
```

In this case, the Go command executes the file in memory as if the following command line had been typed:

```
PROG PARAM1 PARAM2/C
```

Testing and debugging therefore reflect a normal runtime environment for PROG.COM.

OUTPUT

Name: Output

Purpose: Sends the <*byte*> specified to the output port specified by <*value*>.

Syntax: O <*value*> <*byte*>

Comments: A 16-bit port address is allowed.

Example: Type:

```
O2F8 4F
```

DEBUG outputs the byte value 4F to output port 2F8.

QUIT

Name: \ Quit

Purpose: Terminates the DEBUG utility.

Syntax: Q

Comments: The Q command takes no parameters and exits DEBUG without saving the file currently being operated on. You are returned to the MS-DOS command level.

Example: To end the debugging session, type:

```
Q <ENTER>
```

DEBUG has been terminated, and control returns to the MS-DOS command level.

REGISTER

Name: Register

Purpose: Displays the contents of one or more CPU registers.

Syntax: R [<*register-name*>]

Comments: If no <*register-name*> is typed, the R command dumps the register save area and displays the contents of all registers and flags.

If a register name is typed, the 16-bit value of that register is displayed in hexadecimal, and then a colon appears as a prompt. You then either type a <*value*> to change the register, or simply press the <ENTER> key if no change is wanted.

The only valid <*register-name*>s are:

AX	BP	SS	
BX	SI	CS	
CX	DI	IP	(IP and PC both refer to the
DX	DS	PC	Instruction Pointer.)
SP	ES	F	

Any other entry for <*register-name*>, DEBUG displays each flag with a two-character alphabetic code. To alter any flag, type the opposite two-letter code. The flags are either set or cleared.

The flags are listed below with their codes for SET and CLEAR:

FLAG NAME	SET	CLEAR
Overflow	OV	NV
Direction	DN Decrement	UP Increment
Interrupt	EI Enabled	DI Disabled
Sign	NG Negative	PL Plus
Zero	ZR	NZ
Auxiliary Carry	AC	NA
Parity	PE Even	PO Odd
Carry	CY	NC

Whenever you type the command RF, the flags are displayed in the order shown above in a row at the beginning of a line. At the end of the list of flags, DEBUG displays a hyphen (-). You may enter new flag values as alphabetic pairs. The new flag values can be entered in any order. You do not have to leave spaces between the flag entries. To exit the R command, press the <ENTER> key. Flags for which new values were not entered remain unchanged.

If more than one value is entered for a flag, DEBUG returns a DF Error message. If you enter a flag code other than those shown above, DEBUG returns a BF Error message. In both cases, the flags up to the error in the list are changed; flags at and after the error are not.

At startup, the segment registers are set to the bottom of free memory, the Instruction Pointer is set to 0100H, all flags are cleared, and the remaining registers are set to zero.

Example: Type:

```
R
```

DEBUG displays all registers, flags, and the decoded instruction for the current location. If the location is CS:11A, then the display will look similar to this:

```
AX=0E00 BX=00FF CX=007 DX=01FF
SP=039D BP=0000 SI=005C DI=0000
DS=04BA ES=04BA SS=04BA CS=04BA
IP=011A NV UP DI NG NZ AC PE NC
04BA:011A  CD21   INT  21
```

If you type:

```
RF
```

DEBUG will display the flags:

```
NV UP DI NG NZ AC PE NC -_
```

Now, type any valid flag designation, in any order, with or without spaces.

For example:

```
NV UP DI NG NZ AC PE NC -
PLEICY <ENTER>
```

DEBUG responds only with the DEBUG prompt. To see the changes, type either the R or RF command:

```
RF
NV UP EI PL NZ AC PE CY-_
```

Press <ENTER> to leave the flags this way,or to specify different flag values.

SEARCH

Name: Search

Purpose: Searches the <*range*> specified for the <*list*> of bytes specified.

Syntax: S <*range*> <*list*>

Comments: The <*list*> may contain one or more bytes, each separated by a space or comma. If the <*list*> contains more than one byte, only the first address of the byte string is returned. If the <*list*> contains only one byte, all addresses of the byte in the <*range*> are displayed.

Example: If you type:

```
SCS:100 110 41
```

DEBUG will display a response similar to this:

```
04BA:0104
04BA:010D
-type:
```

TRACE

Name: Trace

Purpose: Executes one instruction and displays the contents of all registers and flags, and the decoded instruction.

Syntax: T [= <*address*>] [<*value*>]

Comments: If the optional = <*address*> is typed, tracing occurs at the = <*address*> specified. The optional <*value*> causes DEBUG to execute and trace the number of steps specified by <*value*>.

The T command uses the hardware trace mode of the 8086 or 8088 microprocessor. Consequently, you may also trace instructions stored in ROM (Read Only Memory).

Example: Type:

```
T
```

DEBUG returns a display of the register, flags, and decoded instruction for that one instruction. Assume that the current position is 04BA:011A; DEBUG might return the display:

```
AX=0E00 BX=00FF CX=0007 DX=01FF
SP=039D BP=0000 SI=005C DI=0000
DS=04BA ES=04BA SS=04BA CS=04BA
IP=011A NV UP DI NG NZ AC PE NC
04BA:011A  CD21    INT  21
```

If you type

```
T=011A 10
```

DEBUG executes sixteen (10 hex) instructions beginning at 011A in the current segment, and then displays all registers and flags for each instruction as it is executed. The display scrolls away until the last instruction is executed. Then the display stops, and you can see the register and flag values for the last few instructions performed. Remember that <CONTROL-S> suspends the display at any point, so that you can study the registers and flags for any instruction.

UNASSEMBLE

Name: Unassemble

Purpose: Disassembles bytes and displays the source statements that correspond to them, with addresses and byte values.

Syntax: U [<*range*>]

Comments: The display of disassembled code looks like a listing for an assembled file. If you type the U command without parameters, 20 hexadecimal bytes are disassembled at the first address after that displayed by the previous Unassemble command. If you type the U command with the <*range*> parameter, then DEBUG disassembles all bytes in the range. If the <*range*> is given as an <*address*> only, then 20H bytes are disassembled instead of 80H.

Example: Type:

```
U04BA:100 L10
```

DEBUG disassembles 16 bytes beginning st address 04BA:0100:

```
04BA:0100   206472    AND   [SI + 72],AH
04BA:0103   69        DB    69
04BA:0104   7665      JBE   016B
04BA:0106   207370    AND   [BP + DI + 70],DH
04BA:0109   65        DB    65
04BA:L010A  63        DB    63
04BA:010B   69        DB    69
04BA:010C   66        DB    66
04BA:010D   69        DB    69
04BA:010E   63        DB    63
04BA:010F   61        DB    61
```

If you type

```
U04ba:0100 0108
```

The display will show:

```
04BA:0100  206472      AND  [SI+72],AH
04BA:0103  69          DB   69
04BA:0104  7665        JBE  016B
04BA:0106  207370      AND  [BP+DI+70],DH
```

If the bytes in some addresses are altered, the disassembler alters the instruction statements. The U command can be typed for the changed locations, the new instructions viewed, and the disassembled code used to edit the source file.

WRITE

Name: Write

Purpose: Writes the file being debugged to a disk file.

Syntax: W [<*address*> [<*drive*> <*record*> <*record*>]]

Comments: If you type W with no parameters, BX:CX must already be set to the number of bytes to be written; the file is written beginning from CS:100. If the W command is typed with just an address, then the file is written beginning at that address. If a G or T command has been used, BX:CX must be reset before using the Write command without parameters. Note that if a file is loaded and modified, the name, length, and starting address are all set correctly to save the modified file (as long as the length has not changed).

The file must have been named either with the DEBUG invocation command or with the N command (refer to the Name command earlier in this manual). Both the DEBUG invocation and the N command format a filename properly in the normal format of a file control block at CS:5C.

If the W command is typed with parameters, the write begins from the memory address specified; the file is written to the <*drive*> specified (the drive designation is numeric here — 0 = A:, 1 = B:); DEBUG writes the file beginning at the logical record number specified by the first <*record*>; DEBUG continues to write the file until the number of sectors specified in the second <*record*> have been written.

WARNING

Writing to absolute sectors is *EXTREMELY* dangerous because the process bypasses the file handler.

Example: Type:

```
W
```

DEBUG will write the file to disk and then display the DEBUG prompt. Two exmples are shown below.

```
W
-_
```

```
WCS:100 1 37 2B
```

DEBUG writes out the contents of memory, beginning with the address CS:100 to the disk in drive B:. The data written out starts in disk logical record number 37H and consists of 2BH records. When the write is complete, DEBUG displays the prompt:

```
WCS:100 1 37 2B
-_
```

9.5 ERROR MESSAGES

During the DEBUG session, you may receive any of the following error messages. Each error terminates the DEBUG command under which it occurred, but does not terminate DEBUG itself.

ERROR CODE	DEFINITION
BF	Bad flag You attempted to alter a flag, but the characters typed were not one of the acceptable pairs of flag values. See the Register command for the list of acceptable flag entries.
BP	Too many breakpoints You specified more than ten breakpoints as parameters to the G command. Retype the Go command with ten or fewer breakpoints.
BR	Bad register You typed the R command with an invalid register name. See the Register command for the list of valid register names.
DF	Double flag You typed two values for one flag. You may specify a flag value only once per RF command.

CHAPTER 10

USING THE HARD DISK

Introduction
Preparing Your Hard Disk
Creating the MS-DOS Partition (Option 1)
  MS-DOS Only System
  Other Operating System
Activating the Startup Partition (Option 2)
Deleting the MS-DOS Partition (Option 3)
Displaying Partition Information (Option 4)
Systems with Multiple Hard Disk Drives (Option 5)

10.1 INTRODUCTION

The steps that must be taken to prepare the hard disk for use by MS-DOS are described in this chapter. The following error message will appear if the proper steps aren't taken:

Invalid drive specification

It is possible to divide the hard disk into as many as four separate areas (partitions). Partitions can be different sizes. They are created with the hard disk preparation program (FDISK) provided by MS-DOS.

It is possible to specify a partition on the hard disk as the startup partition. This means that the system will load with the operating system in the startup partition, if the disk drive clamp button is not pushed in.

Hard disk drives are assigned letter specifier in the same manner as floppy disk drives. The disk drive letter specifier must be used when you want to save information onto or transfer information from the hard disk. MS-DOS makes the first hard disk drive in your system drive C, regardless of whether you have one or two such drive in your system. If you have a second hard disk drive in your system, that becomes drive D.

10.2 PREPARING YOUR HARD DISK

The MS-DOS hard disk preparation program (FDISK) can be used to create, change, or erase the MS-DOS partition. (A partition for use by another operating system can be created, changed, or erased only by using that operating system's hard disk preparation program.)

One MS-DOS partition can be created on a hard disk at the location and of the size you desire. Erase the MS-DOS partition of, for example, you want to create it again with a different size or at a different location on the disk, If this is the case, you might want to use the BACKUP command to back up your files in your MS-DOS partition first.

The MS-DOS hard disk preparation program (FDISK) supports the following functions:

1. MS-DOS partition creation

2. Activation of the startup partition (the one that will take control then the system is started)

3. MS-DOS partition deletion

4. Hard disk partition information display

These functions are listed in the FDISK menu. The following section describes the procedure for each function.

Start the FDISK program as follows:

Type:

d:FDISK

where d: is the drive where the FDISK program resides. Simply type the following if FDISK is on the default drive:

FDISK

Press <ENTER>. Thefollowing menu will appear:

```
Fixed Disk Setup Program Version 2.00

<FDISK Menu>

Choose one of these options:

    1. Create MS-DOS Partition
    2. Activate Partition
    3. Delete MS-DOS Partition
    4. Display Partition Information

Enter your choice; [1]
```

Type the number of the option required, then press <ENTER>. Option 1 is the default and will be selected automatically if an option number is not typed before pressing <ENTER>. Proceed to the section that describes the option you have selected.

10.3 CREATING THE MS-DOS PARTITION (OPTION 1)

FDISK allows you to create a startup partition on any hard disk. MS-DOS (or any other operating system), however, can only be started from the first hard disk.

Before creating an MS-DOS partition, determine where it should be located and how large it should be. Only one MS-DOS partition can be created on a hard disk if more than one MS-DOS partition is attempted to be created, an error message will appear.

A hard disk is divided into cylinders. The number of cylinders and their sizes can vary depending on the hard disk attached to your computer.

When you select Option 1, the following display will appear:

```
Create MS-DOS Partition

Do you want to use the entire fixed
disk drive for MS-DOS (Y/N) .....?[Y]
```

10.3.1 MS-DOS Only System

1. If MS-DOS is the only operating system to be used with the hard disk, press <ENTER> to accept the default response.

   Then, FDISK will create the MS-DOS partition. The following message will appear:

   ```
   System will be rebooted:

   Insert MS-DOS diskette in drive A:
   Press any key when ready...
   ```

2. With the MS-DOS disk in drive A, press any key to restart MS-DOS. Then, MS-DOS will recognize your hard disk and assign it a drive letter.

The MS-DOS partition has now been created on your hard disk, but before MS-DOS can use it, a directory and other information must be created in the partition. Use the FORMAT command to do this. For example:

At system prompt A>, type

FORMAT C:/S

This formats the MS-DOS partition you have just created. At the same time, the MS-DOS system files are copied onto the MS-DOS partition.

To copy all the programs on the MS-DOS disk to this hard disk, use the COPY command.
For example:

COPY *.* C:

This copies all of the programs in MS-DOS disk directory as well as the system files to the MS-DOS partition.

10.3.2 Other Operating System

1. If you intend to use another operating system with your hard disk, respond to the prompt "Do you want to use the entire fixed disk drive for MS-DOS (Y/N)?" by pressing N and <ENTER>.

   The following message will appear:

   Total disk space is xxx cylinders.
   Maximum available disk space is xxx.
   cylinders at cylinder xxx.

2. The following prompt appears:

   Enter partition size.....[xxx]

   The partition size entry defaults to the largest available space on the hard disk. If you want your MS-DOS partition to use the largest available space (represented by xxxx in the prompt) simply press <ENTER>.

3. The next prompt is:

   Enter start cylinder number .....:[xxx]

The starting cylinder address default depends on the partition size you have specified. It is the first cylinder of the first space on the hard disk large enough for the partition. To locate the MS-DOS partition there, press <ENTER>. Otherwise, type the starting cylinder number you prefer and press <ENTER>. The MS-DOS partition will then be created by FDISK.

The partition information on the screen changes to show the new partition.

4. To go back to the main menu, press the Esc key. The MS-DOS partition has now been created on your hard disk, but before it can be used by MS-DOS a directory and other information must be created in the partition. Use the FORMAT command to do this.
For example:

At system prompt A>, type

```
FORMAT C:/S
```

This formats the MS-DOS partition you have just created. At the same time, the MS-DOS system files are copied onto the MS-DOS partition.

To copy all the program files to this hard disk, use the COPY command.
For example:

At system prompt A>, type

```
COPY *.* C:
```

This copies to the MS-DOS partition all of the programs in the MS-DOS disk directory.

If you want the operating system in the partition you have just created to take control when the computer is started, follow the instructions in "10.4 ACTIVATING THE STARTUP PARTITION (OPTION2)".

10.4 ACTIVATING THE STARTUP PARTITION (OPTION 2)

Select Option 2 for a different operating system in another partition to take control when you start the computer.

The following display appears when you choose Option 2:

Activate Partition

Partition	Status	Type	Start	End	Size
1	N	non-DOS	000	049	50
2	N	non-DOS	050	149	100
3	N	MS-DOS	150	304	155

Total disk space is xxx cylinders

Enter the number of the partition you
want to activate.....:[ ]

1. Enter the number of the partition whose operating system you want to take control when the system is started from the hard disk. Press <ENTER>.

   The partition information on the screen changes to show the new startup partition.

2. To return to the main FDISK menu, press the Esc key. Press it again to return to MS-DOS.

3. To start up with the operating system in the newly assigned startup partition, perform the following steps:

   a. Do not push in the clamp button on drive A.

   b. Press and hold the Ctrl key and the Art key, then press the Del key.

Then, the operating system in the startup partition should take control.

NOTE:

Before using a reconfigured hard disk system, remember to reset the system.

10.5 DELETING THE MS-DOS PARTITION (OPTION 3)

NOTE:

All data in the MS-DOS partition is destroyed with this option. Before you begin this procedure, you may want to back up all your MS-DOS file.

Insert an MS-DOS disk and restart the system from disk drive A to continue processing under MS-DOS.

To start up with a system in another hard disk partition, change the startup partition to that partition number before you erase the MS-DOS partition.

1. When you choose Option 3, a screen similar to the following will be displayed.

   Delete MS-DOS Partition

Partition	Status	Type	Start	End	Size
1	N	non-DOS	000	049	50
2	N	non-DOS	050	099	50
3	N	MS-DOS	100	249	150
4	A	non-DOS	250	304	55

   Total disk space is xxx cylinders

   Warning! Data in the MS-DOS partition
   will be destroyed. Do you wish to
   continue the job.....?[N]

2. If all your files have been backed up and you are ready to continue, type Y and press <ENTER>. To cancel the operation, press either <ENTER> or the Esc key to return to the main FDISK menu.

   The partition information displayed on the screen is updated if you type Y and press <ENTER>.

The MS-DOS partition has now been erased. It will be necessary to start up with another operating system from the hard disk, or restart MS-DOS from a disk to proceed.

10.6 DISPLAYING PARTITION INFORMATION (OPTION 4)

This option is used to display hard disk partition information.

When Option 4 is chosen, a similar screen to the following appears:

Display Partition Information

Partition	Status	Type	Start	End	Size
1	A	MS-DOS	000	199	200
2	N	non-DOS	200	199	105

Total disk space is xxx cylinders

For each assigned partition, one line is shown.

The relative number of the partition is shown in the partition column (in the order that it appears on the hard disk).

The Status column shows which partition's system gets control when the system unit is started from the hard disk. That partition's status is shown as A (for active), the others are shown as N (non-sctive).

The Type column shows which partition, if any, is the MS-DOS partition.

The starting and ending cylinder numbers of this partition are shown in the Start and End columns.
The size column shows its size in cylinders.

The total amount of space on the hard disk is shown on the next line.

When you are ready to return to the FDISK option menu, press the Esc key.

10.7 SYSTEMS WITH MULTIPLE HARD DISK DRIVES (OPTION 5)

If you want to use FDISK with another hard disk drive, select Option 5. After Option 5 is selected, the current hard disk number on the FDISK options menu will change. This option is valid only if your system has two or more hard disk drives.

APPENDICES

APPENDIX A
Instructions For Users
With Single-Drive Systems
APPENDIX B
Disk Errors
APPENDIX C
ANSI Escape Sequences
APPENDIX D
How To Configure Your System

APPENDIX A
INSTRUCTIONS FOR USERS WITH SINGLE-DRIVE SYSTEMS

On a single-drive system, you enter the commands as you would on a multi-drive system.

You should think of the single-drive system as having two drives (drive A: and drive B:). But instead of A: and B: representing two physical drives as on the multi-drive system, the A: and B: represnet disks.

If you specify drive B: when the "drive A: disk" was last used, you are prompted to insert the disk for drive B;. For example:

```
A >COPY COMMAND. COM B:
Insert diskette for drive B:
and strike any key when ready
      1 File(s) copied
A>_
```

If you specify drive A: when the "drive B: disk" was last used, you are prompted again to change disks. This time, MS-DOS prompts you to insert the "drive A: disk."

The same procedure is used if a command is executed from a batch file. MS-DOS waits for you to insert the appropriate disk and press any key before it continues. You will be prompted to do this.

NOTE

The letter displayed in the system prompt represents the default drive where MS-DOS looks to find a file whose name is entered without a drive specifier. The letter in the system prompt dose not represent the last disk used.

For example, assume that A: is the default drive. If the last operation performed was DIR B:, MS-DOS believes the "drive B: disk" is still in the drive. However, the system prompt is still A:, because A: is still the default drive. If you type DIR, MS-DOS prompts you for the "drive A: disk" because driveA: is the default drive, and you did not specify another drive in the DIR command.

APPENDIX B
DISK ERRORS

If disk or device error occurs at any time during a command or program, MS-DOS returns an error message in the following format:

```
<yyy> ERROR <I/O action> DRIVE x
Abort, Retry, Ignore ? _
```

In this message, <yyy> may be one of the following:

```
WRITE PROTECT
BAD UNIT
NOT READY
BAD COMMAND
DATA
BAD CALL FORMAT
SEEK
NON-DOS DISK
SECTOR NOT FOUND
NO PAPER
WRITE FAULT
READ FAULT
DISK
GENERAL FAILURE
```

The <I/O-action> may be either of the following:

```
READING
WRITING
```

The drive <x> indicates the drive in which the error has occurred.

MS-DOS waits for you to enter one of the following responses:

I Ignore. Ignore the bad sector and pretend the error did not occur.

A Abort. Terminate the program requesting the disk read or write.

R Retry. Repeat the operation. This response is to be used when the operator has corrected the error (such as with NOT READY or WRITE PROTECT errors).

Usually, you will want to attempt recovery by entering responses in this order:

R (to try again)
A (to terminate program and try a new disk)

One other error message might be related to faulty disk read or write:

FILE ALLOCATION TABLE BAD FOR DRIVE x

This message means that the copy in memory of one of the allocation tables has pointers to nonexistent blocks. Possibly the disk was incorrectly formatted or not formatted before use. If this error persists, the disk is currently unusable and must be formatted prior to use.

APPENDIX C
ANSI ESCAPE SEQUENCES

An ANSI escape sequence is a series of characters (beginning with an escape character or keystroke) that you can use to define functions to MS-DOS. Specifically, you can reassign keys, change graphics functions, and affect cursor movement.

This appendix explains how the ANSI escape sequences are defined for MS-DOS version 2.0. Examples on how to use ANSI escape sequences are included at the end of this appendix.

NOTES:

1. The default value is used when no explicit value or a value of zero is specified.

2. Pn represents "numeric parameter." This is a decimal number specified with ASCII digits.

3. Ps represents "selective parameter." This is any decimal number that is used to select a subfunction. Multiple subfunctions may be selected by separating the parameters with semicolons.

C.1 CURSOR FUNCTIONS

The following escape sequences affect the cursor position on the screen.

CUP - Cursor Position

ESC [Pl; Pc H

HVP - Horizontal & Vertical Position

ESC [Pl; Pc f

CUP and HVP move the cursor to the position specified by the parameters. The first parameter specifies the line number, and the second parameter specifies the column number. The default value is 1. When no parameters are specified, the cursor is moved to the home position.

CUU - Cursor Up

ESC [Pn A

This sequence moves the cursor up one line without changing columns. The value of Pn determines the number of lines moved. The default value for Pn is 1. The CUU sequence is ignored if the cursor is already on the top line.

CUD - Cursor Down

ESC [Pn B

This sequence moves the cursor down one line without changing columns. The value of Pn determines the number of lines moved. The default value for Pn is 1. The CUD sequence is ignored if the cursor is already on the bottom line.

CUF - Cursor Forward

ESC [Pn C

The CUF sequence moves the cursor forward one column without changing lines. The value of Pn determines the number of columns moved. The default value for Pn is 1. The CUF sequence is ignored if the cursor is already in the far right column.

CUB - Cursor Backward

ESC [Pn D

This escape sequence moves the cursor back one column without changing lines. The value of Pn determines the number of columns moved. The default value for Pn is 1. The CUB sequence is ignored if the cursor is already in the far left column.

DSR - Device Status Report

ESC [6n

The console driver will output a CPR sequence (see below) on receipt of the DSR escape sequence.

CPR - Cursor Position Report (from console driver to system)

ESC [Pn; Pn R

The CPR sequence reports current cursor position via standard input. The first parameter specifies the current line and the second parameter specifies the current column.

SCP - Save Cursor Position

ESC [s

The current cursor position is saved. This cursor position can be restored with the RCP sequence (see below).

RCP - Restore Cursor Position

ESC [u

This sequence restores the cursor position to the value it had when the console driver received the SCP sequence.

C.2 ERASING

The following escape sequences affect erase functions.

ED - Erase Display

ESC [2 J

The ED sequence erases the screen, and the cursor goes to the home position.

EL - Erase Line

ESC [K

This sequence erases from the cursor to the end of the line (including the cursor position).

C.3 MODES OF OPERATION

The following escape sequences affect screen graphics.

SGR - Set Graphics Rendition

ESC [Ps; ... ; Ps m

The SGR escape sequence invokes the graphic functions specified by the parameter(s) described below. The graphic functions remain until the next occurrence of an SGR escape sequence.

Parameter	Parameter Function	
0	All Attributes off	
1	Bold on	
4	Underscore on	(monochrome displays only)
5	Blink on	
7	Reverse Video on	
8	Concealed on	(ISO 6429 standard)
30	Black foreground	(ISO 6429 standard)
31	Red foreground	(ISO 6429 standard)
32	Green foreground	(ISO 6429 standard)
33	Yellow foreground	(ISO 6429 standard)
34	Blue foreground	(ISO 6429 standard)
35	Magenta foreground	(ISO 6429 standard)
36	Cyan foreground	(ISO 6429 standard)
37	White foregound	(ISO 6429 standard)
40	Black background	(ISO 6429 standard)
41	Red background	(ISO 6429 standard)
42	Green background	(ISO 6429 standard)
43	Yellow background	(ISO 6429 standard)
44	Blue background	(ISO 6429 standard)
45	Magenta background	(ISO 6429 standard)
46	Cyan background	(ISO 6429 standard)
47	White background	(ISO 6429 standard)

SM - Set Mode

ESC [= Ps h
or ESC [= h
or ESC [= 0 h
or ESC [? 7h

The SM escape sequence changes the screen width or type to one of the following parameters:

Parameter	Parameter Function
0	40 x 25 black and white
1	40 x 25 color
2	80 x 25 black and white
3	80 x 25 color
4	320 x 200 color
5	320 x 200 black and white
6	640 x 200 black and white
7	wrap at end of line

RM - Reset Mode

ESC [= Ps I
or ESC [= I
or ESC [= 0I
or ESC [? 7I

Parameters for RM are the same as for SM (Set Mode), except that parameter 7 will reset the wrap at the end of line mode.

C.4 KEYBOARD REASSIGNMENT

Although not part of the ANSI 3.64-1979 or ISO6429 standard, the following keyboard reassignments are compatible with these standards.

The control sequence is:

ESC [Pn; Pn; ... Pn p
or ESC ["string"; p
or ESC [Pn; "string", Pn ; Pn, ; "string" ; Pn p
or any other combination of strings and decimal numbers

The final code in the control sequence (p) is one reserved for private use by the ANSI 3.64-1979 standard.

The first ASCII code in the control sequence defines which code is being mapped. The remaining numbers define the sequence of ASCII codes generated when this key is intercepted. Note that there is one exception: if the first code in the sequence is zero (NUL), then the first and second code make up an extended ASCII redefinition.

Examples: 1. Reassign the Q and q key to the A and a key (and vice versa):

ESC [6 5 ; 8 1 P	A becomes Q
ESC [9 7 ; 1 1 3 p	a becomes q
ESC [8 1 ; 6 5 p	Q becomes A
ESC [1 1 3 ; 9 7 p	q becomes a

2. Reassign the F10 key to a DIR command followed by a carriage return:

ESC [0 ; 6 8 ; "d i r" ; 1 3 p

The 0;68 is the extended ASCII code for the F10 key; 13 decimal is a carriage return.

APPENDIX D
HOW TO CONFIGURE YOUR SYSTEM

In many cases, there are installation-specific settings for MS-DOS that need to be configured at system startup. An example of this is a standard device driver, such as an online printer.

The MS-DOS configuration file (CONFIG.SYS) allows you to configure your system with a minimum of effort. With this file, you can add device drivers to your system at startup. The configuration file is simply an ASCII file that has certain commands for MS-DOS startup (boot). The boot process is as follows:

1. The disk boot sector is read. This contains enough code to read MS-DOS code and the installation's BIOS (machine-dependent code).

2. The MS-DOS code and BIOS are read.

3. A variety of BIOS initializations are done.

4. A system initialization routine reads the configuration file (CONFIG.SYS), if it exists, to perform device installation and other user options. Its final task is to execute the command interpreter, which finishes the MS-DOS boot process.

D.1 CHANGING THE CONFIG.SYS FILE

If there is not a CONFIG.SYS file on the MS-DOS disk, you can use the MS-DOS editor, EDLIN, to create a file; then save it on the MS-DOS disk in your root directory.

The following is a list of commands for the configuration file CONFIG.SYS:

BUFFERS = <*number*>
: This is the number of sector buffers that will comprise the system list. It is installation-dependent. If not set, 10 is a reasonable number.

FILES = <*number*>
: This is the number of open files that the XENIX system calls can access. It is installation-dependent. If not set, 10 is a reasonable number.

LASTDRIVE = <*alphabetic character*>
: You can specify any alphabetic character A through Z. It corresponds the last valid drive letter that MS-DOS accepts. The default value is E.

DEVICE = <*filename*>
: This installs the device driver in <*filename*> into the system list. (See below.)

BREAK = <**ON or OFF**>
: If ON is specified (the default is OFF), a check for CONTROL-C as input will be made every time the system is called. ON improves the ability to abort porgrams over previous versions of the MS-DOS.

SHELL = <*filename*>
: This begins execution of the shell (top-level command processor) from <*filename*>.

A typical configuration file might look like this:

```
Buffers = 10
Files = 10
Device = \BIN\NETWORK.SYS
Break = ON
Shell = A:\BIN\COMMAND.COM A:\BIN /P
```

Note here that the Buffers and Files parameters are set to 10. The system initialization routine will search for the filename \BIN\NETWORK.SYS to find the device that is being added to the system. This file is usually supplied on disk with your device. Make sure that you save the device file in the pathname that you specify in the Device parameter.

This configuration file also sets the MS-DOS command EXEC to the COMMAND. COM file located on disk A: in the \ BIN directory. The A: \ BIN tells COMMAND.COM where to look for itself when it needs to re-read from disk. The /P tells COMMAND.COM that it is the first program running on the system so that it can process the MS-DOS EXIT command.

D. 2 DEVICE COMMAND

ANSI.SYS

The ANSI.SYS command will extend the control of the screen and keyboard by replacing the present input and output software support. On the DOS system disk, create a file using a text editor with the name CONFIG.SYS and type the command line DEVICE = ANSI.SYS. When the disk is booted DOS will read the CONFIG.SYS file and load ANSI.SYS as a replacement driver for the standard I/O software support.

VDISK.SYS

A useful tool can be created from your computer's memory by using the VDISK.SYS file. VDISK.SYS is a virtual disk (memory simulated disk) which operates just like a floppy disk drive. A discussion of the attributes particular to virtual disks follows.

- A virtual disk gives you fast processing because it operates at the speed of your computer's memory.

- Because virtual disks are in memory, their contents are destroyed when a system re-boot is performed or power loss occurs.

- Virtual disks function similar to floppy disks and are assigned a drive letter. For example, if you have only two floppy disk drives in your system they are labeled drive A and B. The formation of virtual disks would then be designated as drive C, drive D, etc.

- To increase the capacity or the number of virtual disks, memory expansion boards (address starts at the 1 megabyte boundary) can be installed and used to their full capability. Without the use of expansion boards the location of virtual disk buffers will reside in low memory.

- VDISK.SYS will increase the size of DOS by 720 bytes for each designated virtual disk plus the buffer size of each.

- As an identification aid, each virtual disk has a volume label.

- To have control over VDISK.SYS the following parameters can be set:
  Virtual disk size (memory allocation)
  Sector size
  Directory entory number

VDISK Installation

Using a text editor, create a file with the name of CONFIG.SYS and add the following statement to install the VDISK device driver.

DEVICE = [*drive*:] [*pathname*] VDISK.SYS [*bbb*] [*sss*] [*ddd*] [/E]

[*drive*:] [*pathname*] specify the drive and directory path of the VDISK.SYS file.

The first parameter *bbb* designates the size in kilobytes of the virtual disk. A minimum size of 1 to a maximum size of the amount of available memory can be specified using a decimal value. VDISK.SYS automatically performs the following memory adjustments.

- If the value is less than 1 kilobyte or more than available memory, the default size of 64 kilobytes is used.

- If available memory is less than 64 kilobytes when the VDISK parameters are being set, an error message is displayed and the virtual disk is not initialized.

- VDISK will always reduce the specified size of the virtual disk so a minimum of 64 kilobytes of available memory remains.

The second parameter *sss* designates the sector size in bytes. Sector size can only be one of three values, 128, 256, or 512. The default size of 128 is used if the parameter was not specified or an incorrect value had been entered. A 128 byte sector size is recommended for relatively small files to conserve space.

The third parameter *ddd* designates the number of directory entries for a virtual disk. For each file, one directory entry is required. The minimum value is 2 entries and maximum is 512 entries with a default value of 64 entries. VDISK.SYS automatically performs the following adjustments to the value.

- The value is adjusted to a multiple of the sector size. (Each directory entry occupies 32 bytes.) For example, if the directory parameter value is 12, and the specified sector size is 256, 16 directory entries will be available. Since 12 entries $\times 32$ bytes $=$ 384 bytes, and our sector size is 256, two sectors are required or $256 \times 2 = 512$ bytes are allocated for the directory generating 512 bytes $\div 32$ bytes $= 16$ directory entries.

- The directory value is reduced one sector at a time until the virtual disk size can contain the directory, file allocation table and 2 additional sectors. If the conditions are not met and the value is 1 then a error message is displayed and the virtual disk is not initialized.

the last parameter, the /E switch, is set to use the memory expansion boards installed in your system. The expanded memory address at 1 magabyte or above and will contain only the virtual disk buffer, the device driver is contained in low memory. By stating multiple command lines (DEVICE = VDISK.SYS) in the CONFIG.SYS file more than one virtual disk can be created and sequentially arranged from the 1 megabyte boundary. An error message is displayed and the virtual disk is not initialized if expanded memory is not installed.

An example of a command line in the CONFIG.SYS file is,

DEVICE = VDISK.SYS 160 256 32

which would create a virtual disk with a size of 160 kilobytes, 256 byte sectors and 32 directory entries.

When DOS is booted, the message

VDISK Version 1.0 virtual disk x

is displayed to inform you that the virtual disk is being installed. If an error occurs an error message will follow. The x represents the assigned drive letter.

The following messages are useful information that is displayed to verify the virtual disk parameters.

Buffer size: n

Sector size: n

Directory entries: n

n represents a number.

INDEX

+ (command character) (MS-LINK) 8-11
.BAT 4-5
.EXE (MS-LINK) 8-6, 8-13
.EXE file (MS-LINK) 8-1
.LIB (MS-LINK) 8-5
.MAP (MS-LINK) 8-6, 8-13
.OBJ (MS-LINK) 8-5, 8-13
/1 (FORMAT) 5-39
/8 (FORMAT) 5-39
/B (FORMAT) 5-39
/S (FORMAT) 5-38
/V (FORMAT) 5-39
; (command character) (MS-LINK) 8-11
<F1> 7-4
<F2> 7-5
<F3> 7-6
<F4> 7-8
<F5> 7-13
<F6> 7-12
<ESC> 7-9
<DEL> 7-7
<INS> 7-10

Abort B-1
Addresses (MS-LINK) 8-4
ANSI escape sequences C-1
    Cursor funcitons C-2
    Erasing C-4
    Keyboard reassignment C-6
    Modes of opreation C-5
ASSIGN 5-4
AUTOEXEC.BAT 4-7, 5-22
Automatic Program Execution 2-8

Backup 3-6
Basic 4-9
Batch processing 4-5
BREAK 5-8

Change directory 3-14
CHDIR 5-9
CHKDSK 2-11,5-10
Class (MS-LINK) 8-13

Class name (MS-LINK) 8-4
Classes (MS-LINK) 8-3
CLS 5-13
Command Characters
+ (MS-LINK) 8-11
; (MS-LINK) 8-11
Command Characters (MS-LINK) 8-11
Command processor 2-1, 2-3, 4-1
Command Prompts
List File (MS-LINK) 8-8, 8-13
Object Modules (MS-LINK) 8-8, 8-13
Run File (MS-LINK) 8-8, 8-13
Summary of Methods to Start (MS-LINK) 8-7
COMMAND. COM 2-1
Commands
ASSIGN 5-4
ATTRIB 5-5
BACKUP 5-6
BREAK 5-8
CHDIR 5-9
CHKDSK 2-11, 5-10
CLS 5-13
COMP 5-14
COPY 3-5, 5-16
CTTY 5-21
DATE 5-22
DEL 5-24
DIR 2-9, 3-12, 5-25
DISKCOMP 5-27
DISKCOPY 2-6, 5-29
ECHO 5-88
EXE2BIN 5-32
EXIT 5-35
External 4-1
FIND 4-13, 5-36
FOR 5-89
FORMAT 2-4, 2-9, 5-38
GOTO 5-90
GRAPHICS 5-46
IF 5-92
Internal 4-1
JOIN 5-47
LABEL 5-49

MKDIR 3-14, 5-51
MODE 5-52
MORE 4-13, 5-52
PARK 5-56
PATH 5-58
PAUSE 4-5, 5-92
PRINT 5-59
PROMPT 5-63
RECOVER 5-65
REM 4-5, 5-66
REN 5-67
RESTORE 5-68
RMDIR 3-15, 5-70
SET 5-71
SHARE 5-72
SHIFT 4-10, 5-93
SORT 4-13, 5-74
SUBST 5-76
SYS 5-78
TIME 5-80
TREE 5-82
TYPE 5-83
VER 5-84
VERIFY 5-85
VOL 5-86
Compiler 8-1
COMP 5-14
Control characters 6-6
CONTROL-C 4-4
CONTROL-S 4-5
CONTROL-Z 4-9, 4-10
COPY command 3-5
CTTY 5-21
Current date 2-1
Current time 2-2
Cursor functions C-2

Data Error B-1
Data protection 2-6
Date 2-1
DEBUG Commands
Assemble 9-8
Compare 9-11

Dump 9-12
Enter 9-14
Fill 9-16
Go 9-17
Hex 9-19
Input 9-20
Load 9-21
Move 9-23
Name 9-24
Output 9-27
Quit 9-28
Register 9-29
Search 9-32
Trace 9-33
Unassemble 9-35
Write 9-37
DEBUG Errors
BF - Bad flag 9-39
BP - Too many breakpoints 9-39
BR - Bad register 9-39
DF - Double flag 9-39
Default drive 2-3
Delimiters 4-4
DIR 2-9, 3-12
Directories
Hierarchical 3-8
Directory 2-9,3-7
Making 3-14
Removing 3-15
Working 3-7
Disk Error B-1
Disk Errors
Abort B-1
Data Error B-1
Disk Error B-1
File Allocation Table Bad For Drive x B-2
Ignore B-1
Not Ready Error B-1
Retry B-2
Sector Not Found Error B-1

Seek Error B-1
Write Fault Error B-1
Write Protect Error B-1
DISKCOPY 2-6, 5-29
Disks
Backup 2-6
Drive designation 3-1, 4-3
Drive designations (MS-LINK) 8-5
Dummy parameters 4-10

ECHO 5-88
Editing commands
Replace mode 7-12
Copy characters 7-4
Copy template 7-6
Enter/exit insert mode 7-10
Enter replace mode 7-12
New template 7-13
Quit 7-9
Quit input 7-9
Skip multiple characters 7-8
Skip one character 7-7
EDLIN Commands
Append Lines 7-19
Copy Lines 7-20
Delete Lines 7-22
Edit Line 7-24
End Editing 7-26
Insert Text 7-27
List Text 7-30
Move 7-33
Page 7-34
Quit 7-35
Replace Text 7-36
Search Text 7-40
Transfer Text 7-43
Write Lines 7-44
EDLIN Errors
Bad Parameter 7-47
Cannot edit .BAK file—rename file 7-45
Destination Number 7-47
Disk Full 7-46
Entry error 7-45
File Creation 7-48
File Not Found 7-47

Filename Not Specified 7-47
Insufficient Memory 7-47
Invalid Drive 7-46
Line too long 7-46
Merge Room 7-48
No room in directory for file 7-45
Version Error 7-46
Erasing C-4
EXIT 5-35
External commands 3-11
External reference (MS-LINK) 8-1

FDISK options
Activate startup partition 10-6
Change the current hard disk number 10-9
Create MS-DOS partition 10-3
Delete MS-DOS partition 10-7
Display partition information 10-8
File Allocation Table 2-9
File Allocation Table Bad For Drive x B-2
File specification 3-2
File system 3-7
Filename 3-1, 4-3
Filename extension 3-1, 4-3
.COM 4-2
.EXE 4-2
Filename extensions - default
.EXE (MS-LINK) 8-6
.LIB (MS-LINK) 8-5
.MAP (MS-LINK) 8-6
.OBJ (MS-LINK) 8-5
Filename extensions - default (MS-LINK) 8-5
Filenames
Illegal 3-4
Files 2-8
Files that MS-LINK uses (MS-LINK) 8-5
Filespec 4-3
Filters 4-13
FIND 4-13, 5-36
FOR 5-89
FORMAT 2-4
FORMAT command 2-4

GOTO 5-90
Groups (MS-LINK) 8-3

Hidden files 2-11
Hierarchical directory 3-8

Ignore B-1
Illegal filenames 3-4
Input 4-12
    Redirection 4-12
Internal commands 3-12

Keyboard reassignment C-6

Loading MS-DOS 2-1

MKDIR 3-14, 5-51
Modes of operation C-5
MORE 4-13, 5-55
Multiplan 4-7

Not Ready Error B-1

Object modules 8-1
Options 4-3
Output 4-12
    Redirection 4-12
Overviews
    MS-LINK 8-1

Parameter 4-10
Parameters
    Replaceable 4-10
Parent directory 3-11, 5-9
Partition 10-1
PATH 5-58
Pathing 3-11
Pathname 3-10, 3-14
PAUSE 4-5
Pipes 4-13
Piping 4-13
PRINT 5-59
PROMPT 5-63

RECOVER 5-65
Relocatable load module (MS-LINK) 8-1
REM 4-5
RENAME(synonym for REN) 5-67
Replaceable parameters 4-10

Reserved filenames
COM 3-4
CON 3-4
LPT 3-4
NUL 3-4
PRN 3-4
Response file (MS-LINK) 8-12
Retry B-2
RMDIR 3-15,5-70
Root directory 2-9

Sector Not Found Error B-1
Seek Error B-1
Segment name (MS-LINK) 8-4
Segments (MS-LINK) 8-4,8-13
Separators 2-2
SET 5-71
SHIFT 4-10,5-93
Shorthand notation 3-11
SORT 4-13,5-74
Source code 8-1
Source drives 4-5
Special characters 3-2
Summary of Methods to Start (MS-LINK) 8-7
Subdirectory 3-13
MS-LINK
/DSALLOCATE (MS-LINK) 8-15
/HIGH (MS-LINK) 8-15
/LINENUMBERS (MS-LINK) 8-16
/MAP (MS-LINK) 8-16
/NO (MS-LINK) 8-17
/PAUSE (MS-LINK) 8-16
/STACK (MS-LINK) 8-17

Time 2-1

VER 5-84
VERIFY 5-85
VM.TMP (MS-LINK) 8-6
VOL 5-86

Wild cards 3-2,4-4
The * wild card 3-3
The ? wild card 3-3
Working directory 3-8,3-11,5-9
Displaying 3-13
Write Fault Error B-1
Write Protect Error B-1